明 細 書

キャッシュメモリ及びその制御方法

技術分野

[0001] 本発明は、プロセッサのメモリアクセスを高速化するためのキャッシュメモリ及びその制御方法に関する。

背景技術

[0002] 近年のマイクロプロセッサでは、例えば、SRAM(Static Random Access Memory)等から成る小容量で高速なキャッシュメモリをマイクロプロセッサの内部、もしくはその近傍に配置し、データの一部をキャッシュメモリに記憶することによって、マイクロプロセッサのメモリアクセスを高速化させている。

[0003] キャッシュの効率向上(ヒット率向上、キャッシュミスレイテンシ低減)のため、キャッシュミスが発生する前に、近い未来に使用するデータを予めキャッシュにフィルする技術としてプリロード(又はプリフェッチ)がある(例えば、特許文献1)。

[0004] 従来のプリフェッチ技術では、プリフェッチ命令により指定したアドレスを含むラインをキャッシュにロードしている。これにより、キャッシュミスの低減を図っている。

特許文献1:特開平7−295882号公報

発明の開示

発明が解決しようとする課題

[0005] しかしながら、上記従来技術によれば、例えば、ソフトウェアによって、ループの外側で一括してプリフェッチを行った場合、ループで必要なデータ領域をあらかじめすべてキャッシュ上に確保することになるため、キャッシュの容量が小さい場合には、それ以外の必要なデータがキャッシュから追い出され、キャッシュミスが発生する。また、データのキャッシング終了、無効化等をループの外側で一括して行う場合は、ループを抜けるまでこれらの動作によるキャッシュの開放が行われないため、キャッシュの容量が不足し、キャッシュミスが発生する。

[0006] また、ループ中にソフトウェアによってキャッシュ操作を行う命令を挿入した場合、キャッシュ操作を行うアドレスをループの中でソフトウェアによって管理する必要がある。

つまり、キャッシュ操作を行う命令をループの中に記述する必要があるため、性能の劣化が発生する。

[0007] さらに、メモリへのアクセスの状況をハードウェアによって監視し、ハードウェアによって自動的にキャッシュ操作をする場合、正確な予測が行えないと無駄な転送が発生する、あるいは、ソフトウェアからの正確な情報がないとキャッシュ上のデータと外部メモリ上のデータの整合性が取ることができないため、ハードウェアの予測によってこれらの操作を行うことは困難である。

課題を解決するための手段

[0008] 本発明の目的は、プロセッサからあらかじめ設定された情報を元に、ハードウェアがプロセッサのプログラム動作の状況を監視し、同期を取りつつ適切なタイミングでキャッシュに対して操作を行うキャッシュメモリシステムを提供することにある。

[0009] 上記目的を達成するため本発明のキャッシュメモリシステムは、プロセッサの状態に関する条件を生成する条件生成手段と、現在のプロセッサの状態が前記条件を満たすかどうかを判定する判定手段と、操作対象となるアドレスを生成するアドレス生成手段と、前記判定手段が条件を満たすと判定したときに前記アドレス生成手段によって生成されたアドレスを用いてキャッシュを操作する操作手段とを備える。

[0010] ここで、前記条件生成手段は、前記判定手段が条件を満たすと判定した場合に新たな条件を生成するようにしてもよい。

[0011] この構成によれば、プロセッサの動作状態が条件を満たすようになったときにキャッシュを操作するので、プロセッサの動作の進行状況に同期してキャッシュメモリを操作することができる。またソフトウェア的に実現しないのでプロセッサに負荷をかけることなくキャッシュメモリを効率よく性能劣化を招くことなく動作させることができる。

[0012] ここで、前記条件生成手段は、プロセッサ内の特定レジスタの値に関する条件を生成するようにしてもよい。前記特定レジスタはプログラムカウンタであってもよい。

[0013] この構成によれば、メモリアクセスアドレスや、プログラムフェッチアドレスを前記条件として、プロセッサの状態を監視することができる。

[0014] ここで、前記条件生成手段は、特定のアドレス範囲内へのメモリアクセスおよび特定のアドレス範囲外へのメモリアクセスの何れかを前記条件として生成するようにしても

よい。

[0015] また、前記条件生成手段は、プロセッサが特定命令を実行することを前記条件として生成するようにしてもよい。

[0016] ここで、前記条件生成手段は、現在の条件に特定の演算を施すことによって前記新たな条件を生成するようにしてもよい。

[0017] また、前記条件生成手段はメモリアクセスアドレスを条件として生成し、前記判定手段が条件を満たすと判定した場合に現在の条件に定数を加算することによって前記新たな条件を生成するようにしてもよい。

[0018] ここで、前記定数は、プロセッサにより実行されるポストインクリメント付きロード／ストア命令におけるインクリメント値またはデクリメント値、およびプロセッサにより実行される2回のロード／ストア命令におけるアドレスの差分値の何れかであるようにしてもよい。

[0019] ここで、前記条件生成手段は複数の条件を生成し、前記判定手段は、複数の条件のすべてを満たすかどうかを判定するようにしてもよい。

[0020] また、前記条件生成手段は複数の条件を生成し、前記判定手段は、複数の条件の何れかを満たすかどうかを判定するようにしてもよい。

[0021] ここで、前記操作手段は、前記判定手段が条件を満たすと判定したときに、前記アドレス生成手段により生成されたアドレスに対応するデータがキャッシュに格納されているかどうかを判定するデータ判定手段と、格納されていないと判定された場合に、キャッシュメモリ中のラインを選択する選択手段と、前記選択されたラインが有効でダーディならライトバックを行うライトバック手段と、前記アドレスに対応するデータをメモリからライトバック後の選択されたラインへ転送する転送手段と、前記アドレスをタグとして前記選択されたラインへ登録する登録手段とを備えるようこうせいしてもよい。

[0022] この構成によれば、キャッシュメモリのプリフェッチを、プロセッサの動作状況を監視して同期を取りつつ適切なタイミングで行うことができる。

[0023] ここで、前記操作手段は、前記判定手段が条件を満たすと判定したときに、前記アドレス生成手段により生成されたアドレスに対応するデータがキャッシュに格納されているかどうかを判定するデータ判定手段と、格納されていないと判定された場合に、

キャッシュメモリ中のラインを選択する選択手段と、選択されたラインが有効でダーティであれば、ライトバックを行うライトバック手段と、メモリから選択されたラインへデータを転送することなく、前記生成したアドレスをタグとして選択されたラインへ登録する登録手段とを備える構成としてもよい。

[0024] この構成によれば、キャッシュメモリのラインにデータを転送するとなく確保すること(タッチと呼ぶ)を、プロセッサの動作状況を監視して同期を取りつつ適切なタイミングで行うことができる。

[0025] ここで、前記操作手段は、前記判定手段が条件を満たすと判定したときに、前記アドレス生成手段により生成されたアドレスに対応するデータがキャッシュに格納されているかどうかを判定するデータ判定手段と、格納されていると判定された場合に、キャッシュメモリ中の格納先のラインを選択する選択手段と、選択されたラインが有効でかつダーディであればライトバックを行うライトバック手段とを備える構成としてもよい。

[0026] この構成によれば、キャッシュメモリのラインデータのライトバックを(キャッシング終了)を、プロセッサの動作状況を監視して同期を取りつつ適切なタイミングで行うことができる。

[0027] ここで、前記操作手段は、前記判定手段が条件を満たすと判定したときに、前記アドレス生成手段により生成されたアドレスに対応するデータがキャッシュに格納されているかどうかを判定するデータ判定手段と、格納されていると判定された場合に、キャッシュメモリ中の格納先のラインを選択する選択手段と、選択されたラインを無効化する無効化手段とを備える構成としてもよい。

[0028] この構成によれば、キャッシュメモリのラインの無効化を、プロセッサの動作状況を監視して同期を取りつつ適切なタイミングで行うことができる。

[0029] ここで、前記操作手段は、前記判定手段が条件を満たすと判定したときに、前記アドレス生成手段により生成されたアドレスに対応するデータがキャッシュに格納されているかどうかを判定するデータ判定手段と、格納されていると判定された場合に、キャッシュメモリ中の格納先のラインを選択する選択手段と、ラインのアクセス順序を示す順序情報に対して、選択されたラインのアクセス順序を変更する変更手段とを備える構成としてもよい。

[0030] この構成によれば、キャッシュメモリのラインのアクセス順序情報を、プロセッサの動作状況を監視して同期を取りつつ適切なタイミングで行うことができる。これによりいわゆるLRUによるキャッシュのリプレース順序を早めたり遅くしたりすることができる。

[0031] ここで、前記条件生成手段により前記条件としてメモリアドレスを生成し、前記操作手段は、さらに、前記条件生成手段により生成されたメモリアドレスがラインの途中を指す場合に、当該ラインの先頭、次のラインの先頭および前のラインの先頭の何れかを指すように調整することによりアドレスを生成する調整手段を備える構成としてもよい。

[0032] この構成によれば、プロセッサがアドレス昇順にアクセスする場合も、アドレス高順にアクセスする場合も、次に必要なラインのアドレスを適切に算出することができる。

[0033] また、本発明のキャッシュメモリの制御方法についても上記と同様の手段、作用を有する。

[0034] 本発明のキャッシュメモリによれば、プロセッサの動作の進行状況に同期してキャッシュメモリを操作することができる。またソフトウェア的に実現しないのでプロセッサに負荷をかけることなくキャッシュメモリを効率よく性能劣化を招くことなく動作させることができる。

[0035] 例えば、プロセッサがポストインクリメント付きロード命令によってシーケンシャルにメモリアクセスする場合には、効率よくプリフェッチすることができる。また、プロセッサがポストインクリメント付きストア命令によってシーケンシャルにデータを書き込む場合には、メモリからキャッシュメモリにデータをロードするペナルティを削除することができ、さらに効率よくキャッシュエントリーをタッチ(確保)することができる。

[0036] また、プロセッサがポストインクリメント付きストア命令によってシーケンシャルにデータを書き込む場合に、プロセッサのストア命令の進行状況に応じて、ストアが完了したラインを予測し、予測したラインにキャッシング終了属性や無効化属性を設定するので、プロセッサでは、キャッシュメモリのラインサイズやライン境界を認識しなくても、ライトバックすべきラインをキャッシュメモリにおいて予測し、効率よくクリーニング(ライトバック)や、キャシュエントリーの開放(無効化)を行うことができる。

[0037] さらに、プロセッサがポストインクリメント付きロード命令によってシーケンシャルにデ

ータを読み出す場合に、読み出しが完了したラインデータを保持するキャッシュエントリーにリプレース属性が設定され、真っ先にリプレース対象として選択されるので、アクセス頻度の低いデータがキャッシュメモリに残ることによるキャッシュミスの誘発を低減することができる。

発明の効果

[0038] 本発明のキャッシュメモリによれば、プロセッサの動作の進行状況に同期してキャッシュメモリを操作することができる。またソフトウェア的に実現しないのでプロセッサに負荷をかけることなくキャッシュメモリを効率よく性能劣化を招くことなく動作させることができる。また、メモリアクセスアドレスや、プログラムフェッチアドレスを前記条件として、プロセッサの状態を監視することができる。

[0039] たとえば、キャッシュメモリのラインにデータを転送するとなく確保すること(タッチと呼ぶ)を、プロセッサの動作状況を監視して同期を取りつつ適切なタイミングで行うことができる。

[0040] また、キャッシュメモリのラインデータのライトバックを(キャッシング終了)を、プロセッサの動作状況を監視して同期を取りつつ適切なタイミングで行うことができる。キャッシュメモリのラインの無効化を、プロセッサの動作状況を監視して同期を取りつつ適切なタイミングで行うことができる。

[0041] さらに、キャッシュメモリのラインのアクセス順序情報を、プロセッサの動作状況を監視して同期を取りつつ適切なタイミングで行うことができる。これによりいわゆるLRUによるキャッシュのリプレース順序を早めたり遅くしたりすることができる。

[0042] また、プロセッサがアドレス昇順にアクセスする場合も、アドレス高順にアクセスする場合も、次に必要なラインのアドレスを適切に算出することができる。

図面の簡単な説明

[0043] [図1]図1は、本発明の実施の形態1におけるプロセッサ、キャッシュメモリ、メモリを含むシステムの概略構成を示すブロック図である。

[図2]図2は、キャッシュメモリの構成例を示すブロック図である。

[図3]図3は、予測処理部の構成例を示すブロック図である。

[図4]図4(a)は、スタートアドレスレジスタにスタートアドレスを書き込む命令の一例を

示す。図4(b)は、サイズレジスタにサイズを書き込む命令の一例を示す。図4(c)は、コマンドレジスタにコマンドを書き込む命令の一例を示す。図4(d)は、コマンドの一例を示す。

[図5]図5は、スタートアライナ及びエンドアライナの説明図である。

[図6]図6(a)は、プリフェッチ部による予測プリフェッチの説明図である。図6(b)は、タッチ部による予測タッチの説明図である。

[図7]図7は、プリフェッチ部における予測プリフェッチ処理の一例を示すフローチャートである。

[図8]図8は、タッチ部における予測タッチ処理の一例を示すフローチャートである。

[図9]図9は、本発明の実施の形態2におけるキャッシュメモリの構成を示すブロック図である。

[図10]図10は、キャッシュエントリーのビット構成を示す。

[図11]図11は、制御部による使用フラグの更新例を示す。

[図12]図12(a)は、ウィークフラグが存在しない場合にキャッシュエントリーがリプレースされる様子を示す図である。図12(b)リプレース処理におけるウィークフラグWの役割を示す説明図である。

[図13]図13は、予測処理部の構成を示すブロック図である。

[図14]図14(a)は、C設定部によるCフラグ設定処理の説明図である。図14(b)は、W設定部によるWフラグ設定処理の説明図である。

[図15]図15は、Cフラグ設定部におけるCフラグ設定処理の一例を示すフローチャートである。

[図16]図16は、Wフラグ設定部におけるWフラグ設定処理の一例を示すフローチャートである。

[図17]図17は、制御部におけるクリーニング処理の一例を示すフローチャートである。

[図18]図18は、制御部におけるUフラグ更新処理を示すフローチャートである。

[図19]図19は、制御部におけるリプレース処理を示すフローチャートである。

符号の説明

[0044]

1　プロセッサ

2　メモリ

3　キャッシュメモリ

20　アドレスレジスタ

21　メモリI／F

30　デコーダ

31a～31d　ウェイ

32a～32d　比較器

33a～33d　アンド回路

34　オア回路

35　セレクタ

36　セレクタ

37　デマルチプレクサ

38　制御部

39　予測処理部

131a　ウェイ

131a～131d　ウェイ

131b～131d　ウェイ

138　制御部

139　予測処理部

401　コマンドレジスタ

402　スタートアドレスレジスタ

403　サイズレジスタ

404　加算器

405a、405b　スタートアライナ

406a、406b　エンドアライナ

407　アクセスアドレスレジスタ

408　予測値レジスタ

409　定数レジスタ

410　セレクタ

411　加算器

412　比較器

413　実行部

413a　実行部

414　プリフェッチ部

415　タッチ部

416　C設定部

417　W設定部

発明を実施するための最良の形態

[0045]　(実施の形態1)

＜全体構成＞

図1は、本発明の実施の形態1におけるプロセッサ1、キャッシュメモリ3、メモリ2を含むシステムの概略構成を示すブロック図である。同図のように、本発明のキャッシュメモリ3は、プロセッサ1およびメモリ2を有するシステムに備えられる。

[0046]　キャッシュメモリ3は、プロセッサ1から指定された条件に従って、予測プリフェッチを行うよう構成されている。予測プリフェッチとは、プロセッサ1によるメモリアクセスの進行状況に基づいて次にプリフェッチすべきラインアドレスを予測し、予測したラインアドレスのデータをキャッシュメモリにプリフェッチすることをいう。また、プロセッサ1から指定される条件というのは、アドレス範囲、シーケンシャルアクセスする場合のアドレスのインクリメント値またはデクリメント値、またはアクセス方向(アドレス昇順または降順)等である。予測プリフェッチは、プロセッサがシーケンシャルにメモリアクセスする場合に適している。

[0047]　加えて、キャッシュメモリ3は、プロセッサ1から指定された条件に従って、予測タッチを行うよう構成されている。予測タッチとは、プロセッサ1によるメモリアクセスの進行状況に基づいて次にタッチすべきラインアドレスを予測し、予測したラインアドレスのデータをロードすることなくセットキャッシュエントリーにバリッドフラグとタグとを設定する

ことにより確保することをいう。予測タッチは、配列データなどの演算結果をメモリに順次格納する場合などに適している。

[0048] ＜キャッシュメモリの構成＞

以下、キャッシュメモリ3の具体例として、4ウェイ・セット・アソシエイティブ方式のキャッシュメモリに本発明を適用した場合の構成について説明する。

[0049] 図2は、キャッシュメモリ3の構成例を示すブロック図である。同図のように、キャッシュメモリ3は、アドレスレジスタ20、メモリI／F21、デコーダ30、4つのウェイ31a～31d（以下ウェイ0～3と略す）、4つの比較器32a～32d、4つのアンド回路33a～33d、オア回路34、セレクタ35、36、デマルチプレクサ37、制御部38を備える。

[0050] アドレスレジスタ20は、メモリ2へのアクセスアドレスを保持するレジスタである。このアクセスアドレスは32ビットであるものとする。同図に示すように、アクセスアドレスは、最上位ビットから順に、21ビットのタグアドレス、4ビットのセットインデックス（図中のSI）、5ビットのワードインデックス（図中のWI）を含む。ここで、タグアドレスはウェイにマッピングされるメモリ中の領域（そのサイズはセット数×ブロックである）を指す。この領域のサイズは、タグアドレスよりも下位のアドレスビット（A10～A0）で定まるサイズつまり2kバイトであり、1つのウェイのサイズでもある。セットインデックス（SI）はウェイ0～3に跨る複数セットの1つを指す。このセット数は、セットインデックスが4ビットなので16セットある。タグアドレスおよびセットインデックスで特定されるキャッシュエントリーは、リプレース単位であり、キャッシュメモリに格納されている場合はラインデータ又はラインと呼ばれる。ラインデータのサイズは、セットインデックスよりも下位のアドレスビットで定まるサイズつまり128バイトである。1ワードを4バイトとすると、1ラインデータは32ワードである。ワードインデックス（WI）は、ラインデータを構成する複数ワード中の1ワードを指す。アドレスレジスタ20中の最下位2ビット（A1、A0）は、ワードアクセス時には無視される。

[0051] メモリI／F21は、キャッシュメモリ3からメモリ2へのデータのライトバックや、メモリ2からキャッシュメモリ3へのデータのロード等、キャッシュメモリ3からメモリ2をアクセスするためのI／Fである。

[0052] デコーダ30は、セットインデックスの4ビットをデコードし、4つのウェイ0～3に跨る1

6セット中の1つを選択する。

[0053] 4つのウェイ0～3は、同じ構成を有数する4つのウェイであり、4×2kバイトの容量を有する。各ウェイは、16個のキャッシュエントリーを有する。1つのキャッシュエントリーは、バリッドフラグV、21ビットのタグ、128バイトのラインデータ、ダーティフラグDを有する。タグは21ビットのタグアドレスのコピーである。ラインデータは、タグアドレスおよびセットインデックスにより特定されるブロック中の128バイトデータのコピーである。バリッドフラグVは、当該キャッシュエントリーのデータが有効か否かを示す。ダーティフラグDは、当該キャッシュエントリーにプロセッサから書き込みがあったか否か、つまりサブライン中にキャッシュされたデータが存在するが書き込みによりメモリ中のデータと異なるためメモリに書き戻すことが必要か否かを示す。

[0054] 比較器32aは、アドレスレジスタ20中のタグアドレスと、セットインデックスにより選択されたセットに含まれる4つのタグ中のウェイ0のタグとが一致するか否かを比較する。比較器32b～32cについても、ウェイ31b～31dに対応すること以外は同様である。

[0055] アンド回路33aは、バリッドフラグと比較器32aの比較結果とが一致するか否かを比較する。この比較結果をh0とする。比較結果h0が1である場合は、アドレスレジスタ20中のタグアドレスおよびセットインデックスに対応するラインデータが存在すること、つまりウェイ0においてヒットしたことを意味する。比較結果h0が0である場合は、ミスヒットしたことを意味する。アンド回路33b～33dについても、ウェイ31b～31dに対応すること以外は同様である。その比較結果h1～h3は、ウェイ1～3でヒットしたかミスしたかを意味する。

[0056] オア回路34は、比較結果h0～h3のオアをとる。このオアの結果をhitとする。hitは、キャッシュメモリにヒットしたか否かを示す。

[0057] セレクタ35は、選択されたセットにおけるウェイ0～3のラインデータのうち、ヒットしたウェイのラインデータを選択する。

[0058] セレクタ36は、セレクタ35により選択された32ワードのラインデータにうち、ワードインデックスに示される1ワードを選択する。

[0059] デマルチプレクサ37は、キャッシュエントリーにデータを書き込む際に、ウェイ0～3の1つに書き込みデータを出力する。この書き込みデータはワード単位でよい。

[0060] 制御部38は、予測処理部39を含み、キャッシュメモリ3の全体の制御を行う。予測処理部39は、主としてプリフェッチの制御を行う。

[0061] ＜予測処理部の構成＞

図3は、予測処理部39の構成例を示すブロック図である。同図のように予測処理部39は、コマンドレジスタ401、スタートアドレスレジスタ402、サイズレジスタ403、加算器404、スタートアライナ405a、405b、エンドアライナ406a、406b、アクセスアドレスレジスタ407、予測値レジスタ408、定数レジスタ409、セレクタ410、加算器411、比較器412、実行部413を備える。

[0062] コマンドレジスタ401は、プロセッサ1から直接アクセス可能なレジスタであり、プロセッサ1により書き込まれたコマンドを保持する。図4(c)に、コマンドレジスタ401にコマンドを書き込む命令の一例を示す。この命令は、通常の転送命令(mov命令)であり、ソースオペランドとしてコマンドを、デスティネーションオペランドとしてコマンドレジスタ(CR)401を指定している。図4(d)に、コマンドフォーマットの一例を示す。このコマンドフォーマットは、コマンド内容と定数とからなる。ここで、コマンド内容は、予測プリフェッチコマンドと、予測タッチコマンドとの何れかを表す。定数は、プロセッサ1が、シーケンシャルにメモリアクセスする場合のメモリアクセスアドレスのインクリメント値またはデクリメント値を表し、例えば、＋4、＋8、－4、－8等である。なお、コマンド中の定数の代わりに、アクセス方向(アドレス昇順かアドレス降順か)と絶対値(連続アクセスする場合のアドレスの差分)とを含むようにしてもよい。

[0063] スタートアドレスレジスタ402は、プロセッサ1から直接アクセス可能なレジスタであり、プロセッサ1により書き込まれたスタートアドレスを保持する。このスタートアドレスはCフラグ(キャッシングを終了してよいか否かを示すクリーニングフラグ)を設定すべきアドレス範囲の開始位置を示す。図4(a)に、スタートアドレスレジスタ402にスタートアドレスを書き込む命令の一例を示す。この命令も、図4(c)と同様に通常の転送命令(mov命令)である。

[0064] サイズレジスタ403は、プロセッサ1から直接アクセス可能なレジスタであり、プロセッサ1により書き込まれたサイズを保持する。このサイズは、スタートアドレスからのアドレス範囲を示す。図4(b)に、サイズレジスタ403にサイズを書き込む命令の一例を示

す。この命令も、図4(c)と同様に通常の転送命令(mov命令)である。なお、サイズの単位は、バイト数であっても、ライン数(キャッシュエントリー数)であってもよく、予め定められた単位であればよい。

[0065] 加算器404は、スタートアドレスレジスタ402に保持されたスタートアドレスとサイズレジスタ403に保持されたサイズとを加算する。加算結果は、アドレス範囲の終了位置を指すエンドアドレスである。加算器404は、サイズがバイト数指定の場合はバイトアドレスとして加算し、サイズがライン数指定の場合はラインアドレスとして加算すればよい。

[0066] スタートアライナ405a、405bは、スタートアドレスをライン境界の位置に調整する。スタートアライナ405aはエンドアドレスの方向に、405bはエンドアドレスとは反対の方向に調整する。この調整によりプロセッサ1はラインサイズ及びライン境界とは無関係に任意のアドレスをスタートアドレスとして指定することができる。

[0067] エンドアライナ406a、406bは、エンドアドレスをライン境界の位置に調整する。エンドアライナ4065aはスタートアドレスの方向に、406bはステートアドレスとは反対の方向に調整する。この調整によりプロセッサ1はラインサイズ及びライン境界とは無関係に任意の大きさを上記サイズとして指定することができる。

[0068] 図5に、スタートアライナ405a、405b及びエンドアライナ406a、406bの説明図を示す。同図において、プロセッサ1から指定されたスタートアドレスはラインNの途中の任意の位置を指す。スタートアライナ405aは、次のライン(N+1)の先頭を指すよう調整し、調整後のアドレスをアラインスタートアドレスaとして出力する。スタートアライナ405bは、スタートアドレスのデータを含むラインNの先頭を指すよう調整し、調整後のアドレスをアラインスタートアドレスbとして出力する。アラインスタートアドレスが指すラインをスタートラインと呼ぶ。

[0069] また、エンドアドレスはラインMの途中の任意の位置を指す。エンドアライナ406aは、直前のライン(M−1)の先頭を指すよう調整し、調整後のアドレスをアラインエンドアドレスaとして出力する。エンドアライナ406bは、エンドアドレスのデータを含むラインMの先頭を指すよう調整し、調整後のアドレスをアラインエンドアドレスbとして出力する。アラインエンドアドレスが指すラインをエンドラインと呼ぶ。

[0070] 同図のように、スタートアライナ405a及びエンドアライナ406aはライン単位で内側アラインを行う。スタートアライナ405b及びエンドアライナ406bはライン単位で外側アラインを行う。さらに、ライン単位の外側アラインの後、さらに、サブライン単位の外側アラインと内側アラインが可能である。

[0071] アクセスアドレスレジスタ407は、プロセッサ1からのメモリアクセスアドレスを保持する。

[0072] 予測値レジスタ408は、初期値として、アクセスアドレスレジスタ407のメモリアクセスアドレスと定数レジスタ409の定数とを加算した値を予測値として保持し、以降、プロセッサ1がメモリアクセスを実行したときに、アクセスアドレスレジスタ407のメモリアクセスアドレスと予測値とが一致していれば、予測値と定数レジスタ409の定数とを加算した値を新たな予測値として更新し、一致していなければ、新たな初期値に更新する。

[0073] 定数レジスタ409は、プロセッサ1によるメモリアクセスアドレスのインクリメント値またはデクリメント値を保持する。このインクリメント値(またはデクリメント値)は、プロセッサ1がメモリをシーケンシャルにアクセスする場合のポストインクリメント付きロード／ストア命令のインクリメント値(またはデクリメント値)であり、例えば、図4(d)に示したコマンド中の定数として指定される。

[0074] セレクタ410は、アクセスアドレスレジスタ407のメモリアクセスアドレスと予測値レジスタ408の予測値とが一致していれば、予測値レジスタ408を選択し、一致していなければ、アクセスアドレスレジスタ407を選択する。

[0075] 加算器411は、セレクタ410に選択された予測値またはメモリアクセスアドレスと、定数レジスタ409の定数とを加算する。加算後の値は、新たな予測値又は新たな初期値として予測値レジスタ408に保持される。

[0076] 比較器412は、アクセスアドレスレジスタ407のメモリアクセスアドレスと予測値レジスタ408の予測値とが一致するか否かを判定する。

[0077] 実行部413は、プリフェッチ部414とタッチ部415とを備える。

プリフェッチ部414は、プロセッサのロード命令の進行状況に応じて次にロードされるライン推定し、推定したラインをプリフェッチする予測プリフェッチを行う。

[0078] タッチ部415は、プロセッサのストア命令の進行状況に応じて次にストアされるライン推定し、推定したラインを保持するためのキャッシュエントリーを、メモリからデータをロードすることなく確保する(V＝1の設定とタグの設定とを行う)。

[0079] ＜予測プリフェッチ＞

図6(a)は、プリフェッチ部414による予測プリフェッチの説明図である。同図(a)において、ラインN、N＋1、・・・、N＋nはアドレスが連続するラインを示す。スタートアドレスレジスタ402及びサイズレジスタ403により定まるスタートライン、エンドラインは、それぞれラインN＋1、ラインN＋nであるものとする。また、LD(1)、LD(2)、・・・は、ロード命令によるアクセス位置を示している。

[0080] プリフェッチ部414は、最初のロード命令LD(1)の実行時に、そのメモリアクセスアドレスに定数を加算した値を初期値として予測値レジスタ408に保持させる。2回目のロード命令LD(2)の実行時に、そのメモリアクセスアドレスと予測値レジスタ408の予測値とが一致した場合、プリフェッチ部414は、ポストインクリメント付きロード命令によるシーケンシャルアクセスであるものと推定して、次のラインN＋1をプリフェッチする。もし、次のラインがキャッシュメモリに格納されている場合は、なにもしない。

[0081] プロセッサ1によってポストインクリメント付きロード命令によりシーケンシャルアクセスがなされた場合、予測値はメモリアクセス毎に一致していくことになる。

[0082] このようにして、予測値レジスタ408の予測値がラインNからラインN＋1のアドレスにまで更新され、メモリアクセスアドレスと一致したとき、プリフェッチ部414は、ラインN＋2をプリフェッチする。プリフェッチ部414は、この予測プリフェッチをスタートラインからエンドラインまで実行し、エンドラインのプリフェッチが完了すると予測プロフェッチを終了する。

[0083] ＜予測タッチ＞

図6(b)は、タッチ部415による予測タッチの説明図である。図6(a)と比較して、プロセッサ1がストア命令によりシーケンシャルにアクセスする場合に、そのメモリアクセスアドレスと予測値レジスタ408の予測値とが一致した場合に、ポストインクリメント付きストア命令によるシーケンシャルアクセスと推定し、次のラインN＋1に対応するキャッシュエントリーを確保する点が異なる。つまり、プリフェッチの代わりに、キャッシュエ

ントリーにメモリデータをロードしないでタグ及びバリッドフラグを1に設定する点が異なっている。これ以外は予測プリフェッチと同様なので説明を省略する。

[0084] ＜予測プリフェッチ処理＞

図7は、プリフェッチ部414における予測プリフェッチ処理の一例を示すフローチャートである。

同図において、プリフェッチ部414は、コマンドレジスタ401に予測プリフェッチコマンドが保持されていて(S41)、プロセッサがロード命令を実行したとき(S42)、当該メモリアクセスアドレスと定数とを加算した値を初期値として予測値レジスタ408に設定する(S43)。この最初のロード命令のメモリアクセスアドレスと予測値レジスタの予測値とは当然一致しない。予測値レジスタ408はクリアされているかランダムな値を保持しているからである。

[0085] さらに、プロセッサがロード命令を実行し(S44)、かつメモリアクセスアドレスと予測値レジスタの予測値とが一致する場合には(S45)、プリフェッチ部414は、次ラインのラインアドレスを算出し(S46)、算出したラインアドレスがスタートラインからエンドラインまでのアドレス範囲に属していて(S47)、次ラインがキャッシュメモリにエントリーされていなければ(S48)、次ラインをプリフェッチする(S49)。

[0086] このプリフェッチにおいて、プリフェッチ部414は、LRU方式でリプレース対象のウェイを選択し(S401)、当該ウェイのキャッシュエントリーがダーティであればライトバックし(S402、S403)、次ラインのデータを当該キャッシュエントリーにリフィル(プリフェッチ)する(S404)。

[0087] さらに、プリフェッチ部414は、プリフェッチしたラインがエンドラインであれば(S50)予測プリフェッチ処理を終了する。

[0088] また、上記S45において、メモリアクセスアドレスと予測値レジスタの予測値とが一致しない場合には、メモリアクセスアドレスに定数を加算した値が新たな予測値として予測値レジスタ408に設定される。また、一致しない場合は、なにもしない(メモリアクセス待ちの状態)。

[0089] このように、プリフェッチ部414は、予測プリフェッチにおいて、プロセッサのロード命令の進行状況に応じて次のラインを推定し、推定したラインをプリフェッチするので、

プロセッサ1では、キャッシュメモリのラインサイズやライン境界を認識する必要がなく、つまり、プリフェッチ用のアドレスをプロセッサ1にて管理する必要がない。キャッシュメモリにおいて推定に基づいて効率よくプリフェッチを行うことができる。

[0090] ＜予測タッチ処理＞

図8は、タッチ部415における予測タッチ処理の一例を示すフローチャートである。同図は、図7の予測プリフェッチ処理と比較して、S41、S42、S44、S49、S404の各ステップの代わりにS41a、S42a、S44a、S49a、S404aを有する点が異なっている。これ以外は同じなので異なる点について説明する。

[0091] タッチ部415は、S41aにおいて、予測タッチコマンドがコマンドレジスタ401に保持されているかを判定し、S42a、S44aにおいてストア命令が実行されたか否かを判定する。また、タッチ部415は、S49aにおいてプリフェッチする代わりにタッチする。すなわち、S404aにおいて、リプレース対象のキャッシュエントリーメモリデータをロードしないでタグ及びバリッドフラグを1に設定する。

[0092] このように、タッチ部415は、プロセッサのストア命令の進行状況に応じて次のラインを推定し、推定したラインをタッチする(確保する)ので、プロセッサ1では、キャッシュメモリのラインサイズやライン境界を認識する必要がなく、つまりプリフェッチ用のアドレスをプロセッサ1にて管理する必要がない。キャッシュメモリにおいて推定に基づいて効率よくキャッシュエントリーをタッチすることができる。

[0093] ＜変形例＞

なお、本発明のキャッシュメモリは、上記の実施の形態の構成に限るものではなく、種々の変形が可能である。以下、変形例のいくつかについて説明する。

(1)定数レジスタ409、コマンドにより設定される構成を示したが、(a)デフォルトとしてインクリメント値またはデクリメント値を保持する、(b)複数回のロード／ストア命令における2つのメモリアクセスアドレスの差分を算出し、インクリメント値またはデクリメント値として保持する、(c)プロセッサにより指定されたアドレス方向(アドレス昇順かアドレス降順か)に応じて上記(b)を算出し保持する構成としてもよい。

(2)プリフェッチ部414において、次のラインのさらに次のライン等複数ラインをプリフェッチするようにしてもよい。同様にタッチ部415も、複数ラインをタッチするようにして

もよい。

(3)プリフェッチ部414は、比較器412で複数回一致した場合にプリフェッチを開始するようにしてもよい。タッチ部415も同様である。

(4)上記実施の形態では、4ウェイ・セット・アソシエイティブのキャッシュメモリを例に説明したが、ウェイ数は、いくつでもよい。また、上記実施の形態では、セット数が16である例を説明したが、セット数はいくつでもよい。

(5)上記実施の形態では、セット・アソシエイティブのキャッシュメモリを例に説明したが、フル・アソシエイティブ方式やダイレクトマップ方式のキャッシュメモリであってもよい。

(6)上記実施の形態では、サブラインのサイズをラインのサイズの1／4としているが、1／2、1／8、1／16等他のサイズでもよい。その場合、各キャッシュエントリーは、サブラインと同数のバリッドフラグおよびダーティフラグをそれぞれ保持すればよい。

(7)上記実施の形態におけるサイズレジスタ403は、サイズを保持する代わりにプリフェッチすべき回数を保持する構成としてもよい。この場合には、予測処理部39は実際にプリフェッチした回数をカウントし、カウント値がプリフェッチすべき回数に達したとき、プリフェッチを停止する構成とすればよい。このようにプリフェッチ回数をカウントすることにより、プロセッサにおけるプログラム進行状態を監視することができる。

[0094] (実施の形態2)

第1の実施の形態では、説明した予測プリフェッチ及び予測タッチでは今後アクセスするであろうラインを推定する場合の構成を説明した。本実施の形態では、既にアクセスしたであろうラインを推定して予測クリーニング(ライトバック)や予測ウィーク化(アクセス順位の最弱化)や予測無効化も行う構成について説明する。

[0095] ＜キャッシュメモリの構成＞

図9は、本発明の実施の形態2におけるキャッシュメモリの構成を示すブロック図である。同図のキャッシュメモリは、図2に示した構成と比較して、ウェイ31a～31dの代わりにウェイ131a～131dを備える点と、制御部38の代わりに制御部138を備える点とが異なっている。以下、同じ点は説明を省略して、異なる点を中心に説明する。

[0096] ウェイ131aは、ウェイ31aと比べて、各キャッシュエントリー中に、Cフラグ、Wフラグ

及びUフラグが追加されている点と、ライン単位のバリッドフラグVの代わりにサブライン毎のバリッドフラグV0～V3を有する点と、ライン単位のダーティフラグDの代わりにサブライン毎のダーティフラグD0～D3を有する点とが異なる。ウェイ131b～131dも同様である。

[0097] 図10に、キャッシュエントリーのビット構成を示す。1つのキャッシュエントリーは、バリッドフラグV0～V3、21ビットのタグ、128バイトのラインデータ、ウィークフラグW、使用フラグU及びダーティフラグD0～D3を保持する。

[0098] タグは21ビットのタグアドレスのコピーである。

ラインデータは、タグアドレスおよびセットインデックスにより特定されるブロック中の128バイトデータのコピーであり、32バイトの4つのサブラインからなる。

[0099] バリッドフラグV0～V3は、4つのサブラインに対応し、サブラインが有効か否かを示す。

[0100] Cフラグ(クリーニングフラグ)は、キャッシングを終了してよいかどうかを示すキャッシング終了属性を示す。C＝0は、以降に書き込みがなされる可能性があることを意味する。C＝1は、以降に書き込みがなされないことを意味し、ダーティであればクリーニング(ライトバック)によりキャッシングを終了すべきであることを意味する。

[0101] ウィークフラグWは、プロセッサからのアクセスに関しては、これ以上使用するか否かを示し、キャッシュメモリにおけるリプレース制御に関しては、他のキャッシュエントリーよりも先に追い出してもよい最弱のリプレース対象であることを示す。

[0102] 使用フラグUは、そのキャッシュエントリーにアクセスがあったか否かを示し、LRU方式におけるキャッシュエントリー間のアクセス順序データの代わりに用いられる。より正確には、使用フラグUの1は、アクセスがあったことを、0はないことを意味する。ただし、1つのセット内の4つウェイの使用フラグが全て1になれば、0にリセットされる。別言すれば、使用フラグUは、アクセスされた時期が古いか新しいか2つの相対的な状態を示す。つまり、使用フラグUが1のキャッシュエントリーは、使用フラグが0のキャッシュエントリーよりも新しくアクセスされたことを意味する。

[0103] ダーティフラグD0～D3は、4つのサブラインに対応し、そのサブラインにプロセッサから書き込みがあったか否か、つまりサブライン中にキャッシュされたデータが存在す

るが書き込みによりメモリ中のデータと異なるためメモリに書き戻すことが必要か否かを示す。

[0104] 制御部138は、制御部38と比べて、予測処理部39の代わりに予測処理部139を有する点が異なり、予測によりCフラグを設定する点と、予測によりWフラグを設定する点と、LRU方式におけるアクセス順序情報の代わりに使用フラグUを用いる点とが異なる。

[0105] ＜使用フラグUの説明＞

図11は、制御部138による使用フラグの更新例を示す。同図の上段、中断、下段は、ウェイ0～3に跨るセットNを構成する4つのキャッシュエントリーを示している。4つのキャッシュエントリー右端の1又は0は、それぞれ使用フラグの値である。この4つの使用フラグUをU0～U3と記す。

[0106] 同図上段では(U0～U3)＝(1、0、1、0)であるので、ウェイ0、2のキャッシュエントリーはアクセスがあったことを、ウェイ1、3のキャッシュエントリーはアクセスがないことを意味する。

[0107] この状態で、メモリアクセスがセットN内のウェイ1のキャッシュエントリーにヒットした場合、同図中段に示すように、(U0～U3)＝(1、1、1、0)に更新される。つまり、実線に示すようにウェイ1の使用フラグU1が0から1に更新される。

[0108] さらに、同図中段の状態で、メモリアクセスがセットN内のウェイ3のキャッシュエントリーにヒットした場合、同図下断に示すように、(U0～U3)＝(0、0、0、1)に更新される。つまり、実線に示すようにウェイ3の使用フラグU1が0から1に更新される。加えて、破線に示すようにウェイ3以外の使用フラグU0～U2が1から0に更新される。これにより、ウェイ3のキャシュエントリーが、ウェイ0～2の各キャッシュエントリーよりも新しくアクセスされたことを意味することになる。

[0109] 制御部138は、キャッシュミス時にW＝1のキャッシュエントリーが存在しなければ、使用フラグに基づいてリプレース対象のキャッシュエントリーを決定してリプレースを行う。例えば、制御部138は、図5上段では、ウェイ1とウェイ3の何れかをリプレース対象と決定し、図5中段ではウェイ3をリプレース対象と決定し、図5下段ではウェイ0～2の何れかをリプレース対象と決定する。

[0110] ＜ウィークフラグWの説明＞

図12(a)ウィークフラグが存在しないと仮定した場合の比較例であり、キャッシュエントリーがリプレースされる様子を示す図である。同図においても、図11と同様にウェイ0～3に跨るセットNを構成する4つのキャッシュエントリーを示している。、4つのキャッシュエントリー右端の1又は0は、それぞれ使用フラグの値である。また、データEのみアクセス頻度の低いデータを、データA、B、C、Dはアクセス頻度の高いデータとする。

[0111] 同図(a)の第1段目の状態で、プロセッサ1がデータEにアクセスすると、キャッシュミスが発生する。このキャッシュミスにより、例えば、U＝0のキャッシュエントリーの中からアクセス頻度の高いデータCのキャッシュエントリーがアクセス頻度の低いデータEにリプレースされ、第2段目の状態となる。

[0112] 第2段目の状態で、プロセッサ1がデータCにアクセスすると、キャッシュミスが発生する。このキャッシュミスにより、U＝0のキャッシュエントリーであるアクセス頻度の高いデータDのキャッシュエントリーがアクセス頻度の高いデータCにリプレースされ、第3段目の状態となる。

[0113] 第3段目の状態で、プロセッサ1がデータDにアクセスすると、キャッシュミスが発生する。このキャッシュミスにより、例えば、アクセス頻度の高いデータCのキャッシュエントリーがアクセス頻度の高いデータDにリプレースされ、第3段目の状態となる。

[0114] 同様に、第4段目でも、使用頻度の低いデータEはリプレース対象として選択されないで、キャッシュメモリーに残っている。

[0115] 第5段目の状態で、使用頻度の低いデータEは最も古い(U＝0)であることから、リプレース対象として選択されて、追い出される。

[0116] このように、擬似LRU方式において(通常のLRU方式においても)、アクセス頻度の低いデータEによって、4ウェイの場合は最悪4回のキャッシュミスを誘発する場合がある。

[0117] 図12(b)は、リプレース処理におけるウィークフラグWの役割を示す説明図である。

[0118] 同図(b)の第1段目の状態(同図(a)の第1段目と同じ)で、プロセッサ1がデータEにアクセスすると、キャッシュミスが発生する。このキャッシュミスにより、例えば、U＝0

のキャッシュエントリーの中からアクセス頻度の高いデータCのキャッシュエントリーがアクセス頻度の低いデータEにリプレースされる。このとき、プロセッサ1は、データEのキャッシュエントリーにウィークフラグWを1に設定するものとする。これにより、次のキャッシュミス時にデータEのキャッシュエントリーが真っ先に追い出され、第2段目の状態となる。

[0119] 第2段目の状態で、プロセッサ1がデータCにアクセスすると、キャッシュミスが発生する。このキャッシュミスにより、W＝1のキャッシュエントリーであるアクセス頻度の低いデータEのキャッシュエントリーがリプレース対象として選択され、アクセス頻度の高いデータCにリプレースされ、第3段目の状態となる。

[0120] このように、ウィークフラグWを設けることにより、アクセス頻度の低いデータによるキャッシュミスの誘発を低減することができる。

[0121] ＜予測処理部の構成＞

図13は、予測処理部139の構成を示すブロック図である。同図の予測処理部139は、図3に示した予測処理部39と比較して、実行部413の代わりに実行部413aを備える点が異なっている。実行部413aは、実行部413と比べて、C設定部416、W設定部417が追加されている。

[0122] C設定部416は、プロセッサのストア命令の進行状況に応じてストアが完了した直前のラインを推定し、推定したラインのCフラグを1に設定する。

[0123] W設定部417は、プロセッサのストア命令の進行状況に応じてストアが完了した直前のラインを推定し、推定したラインのWフラグを1に設定する。

[0124] ＜Cフラグ設定処理の説明図＞

図14(a)は、C設定部416によるCフラグ設定処理の説明図である。同図(a)において、ラインN、N＋1、・・・、N＋n、スタートライン、エンドラインは、図6(a)と同様である。

[0125] C設定部416は、プロセッサ1がシーケンシャルにラインN、N＋1、・・・に対してデータをストアしていく場合に、例えば、ラインN＋1に対してストア命令ST(1)、ST(2)が実行され、ST(2)のメモリアクセスアドレスと予測値とが一致した場合に、ラインNに対するシーケンシャルアクセスによるストアが完了したものと推定し、ラインNのCフ

ラグを1に設定する。Cフラグが1に設定されたラインNは、キャッシュミスの発生を待たずにクリーニング処理によりライトバックされる。

[0126] 同様、ラインN＋2へのストア命令実行中に予測値を一致した場合、ラインN＋1に対するシーケンシャルアクセスによるストアが完了したものと推定し、ラインN＋1のCフラグを1に設定する。

[0127] このように、C設定部416は、Cフラグの設定をスタートラインからエンドラインまでの範囲内で実行する。

[0128] ＜Wフラグ設定処理の説明図＞

図14(b)は、W設定部417によるWフラグ設定処理の説明図である。同図(a)と比べて、Cフラグの代わりにWフラグを設定する点のみが異なり、これ以外は同様であるので説明を省略する。

[0129] Wフラグが1に設定されたラインは、キャッシュミスの発生時に真っ先にリプレース対象として選択され、キャッシュメモリから追い出される。

[0130] ＜Cフラグ設定処理フロー＞

図15は、C設定部416におけるCフラグ設定処理の一例を示すフローチャートである。

同図において、C設定部416は、コマンドレジスタ401にCフラグ設定コマンドが保持されていて(S41b)、プロセッサがストア命令を実行したとき(S42b)、当該メモリアクセスアドレスと定数とを加算した値を初期値として予測値レジスタ408に設定する(S43)。この最初のストア命令のメモリアクセスアドレスと予測値レジスタの予測値とは当然一致しない。予測値レジスタ408はクリアされているかランダムな値を保持しているからである。

[0131] さらに、プロセッサがストア命令を実行し(S44b)、かつメモリアクセスアドレスと予測値レジスタの予測値とが一致する場合には(S45)、C設定部416は、直前のラインのラインアドレスを算出し(S46b)、算出したラインアドレスがスタートラインからエンドラインまでのアドレス範囲に属していて(S47)、直前のラインがキャッシュメモリにエントリーされていれば(S48b)、直前のラインのCフラグを1に設定する(S49b)。

[0132] さらに、C設定部416は、直前のラインがエンドラインであれば(S50)Cフラグ設定

処理を終了する。

[0133] このように、C設定部416は、プロセッサのストア命令の進行状況に応じて、ストアが完了した直前のラインを推定し、推定したラインのCフラグを1に設定するので、プロセッサ1では、キャッシュメモリのラインサイズやライン境界を認識する必要がなく、つまり、クリーニングしてもよいラインをラインアドレスとしてプロセッサ1にて管理する必要がない。キャッシュメモリにおいて推定に基づいて効率よくクリーニングを行うことができる。

[0134] ＜Wフラグ設定処理フロー＞

図16は、W設定部417におけるWフラグ設定処理の一例を示すフローチャートである。同図のフローは、図15のCフラグ設定処理と比較して、Cフラグの代わりにWフラグを設定する点のみが異なり、これ以外は同様であるので説明を省略する。

[0135] このように、W設定部417は、プロセッサのストア命令の進行状況に応じて、ストアが完了した直前のラインを推定し、推定したラインのWフラグを1に設定するので、キャッシュメモリにおいて推定に基づいてストアが完了した直前のラインを効率よくリプレース対象とすることができる。その際、プロセッサ1では、キャッシュメモリのラインサイズやライン境界を認識する必要がなく、つまり、リプレースしてもよいラインをラインアドレスとしてプロセッサ1にて管理する必要がない。

[0136] ＜クリーニング処理＞

図17は、制御部138におけるクリーニング処理の一例を示すフローチャートである。

同図のように、制御部138は、ループ1の処理(S900～S913)において、セットインデックス(SI)0～15を順に指定する(S901)ことにより、16個のすべてのセットに対してループ2の処理を行う。ループ2の処理(S900～S913)において、制御部138は、セット内のウェイ0～3のCフラグを読み出す(S903)ことにより、C＝1のキャッシュエントリーを探索する(S904)。ループ3の処理(S905～910)において、制御部138は、C＝1のキャッシュエントリーに対して、サブライン単位のダーティフラグを読み出し(S906)、ダーティであれば(S907)、そのサブランのデータをメモリ2に書き戻し(S908)、当該ダーティフラグを0にリセットする(S909)。このサブラインデータの書

き戻しにおいて、制御部138は、ループ4の処理(S920～S923)のように、空きサイクルにおいて(S920)、1ワードずつ書き戻す(S922)

このように、制御部138は、全てのキャッシュエントリーのCフラグを順にチェックして、C＝1のキャッシュエントリーを探索し、ダーティであればキャッシュメモリ3からメモリ2に書き戻す。

[0137] このように、クリーニング処理では、これ以上書き込みされないことを示すCフラグを有するキャッシュエントリーを、キャッシュミスが発生する前にライトバックするので、キャッシュミス時にはロードペナルティが発生するだけでライトバックペナルティの発生を低減することができる。これによりキャッシュメモリの効率を向上させ、アクセス速度を向上させることができる。

[0138] ＜Uフラグ更新処理＞

図18は、制御部138におけるUフラグ更新処理を示すフローチャートである。同図では、バリッドフラグが0(無効)であるキャッシュエントリーの使用フラグUは0に初期化されているものとする。

[0139] 同図において、制御部138は、キャッシュヒットしたとき(ステップS61)、セットインデックスにより選択されたセットにおけるヒットしたウェイの使用フラグUを1にセットし(ステップS62)、そのセット内の他のウェイの使用フラグUを読み出し(ステップS63)、読み出した使用フラグUが全て1であるか否かを判定し(ステップS64)、全て1でなければ終了し、全て1であれば他のウェイの全ての使用フラグUを0にリセットする(ステップS65)。

[0140] このようにして制御部138は、図11、図12(a)(b)に示した更新例のように、使用フラグUを更新する。

[0141] ＜リプレース処理＞

図19は、制御部138におけるリプレース処理を示すフローチャートである。同図において制御部138は、メモリアクセスがミスしたとき(ステップS91)、セットインデックスにより選択されたセットにおける、4つウェイの使用フラグU及びウィークフラグWを読み出し(ステップS92)、W＝1のウェイが存在するか否かを判定する(ステップS93)。W＝1のウェイが存在しないと判定された場合、U＝0のウェイを1つ選択する(ステ

ップS94)。このとき、使用フラグUが0になっているウェイが複数存在する場合は、制御部138はランダムに1つを選択する。また、W＝1のウェイが存在すると判定された場合、Uフラグの値に関わらずW＝1のウェイを1つ選択する(ステップS95)。このとき、ウィークフラグWが1になっているウェイが複数存在する場合は、制御部138はランダムに1つを選択する。

[0142] さらに、制御部138は、当該セットにおける選択されたウェイのキャッシュエントリーを対象にリプレースし(ステップS96)、リプレース後に当該キャッシュエントリーの使用フラグUを1に、ウィークフラグWを0初期化する(ステップS97)。なお、このときバリッドフラグV、ダーティフラグDは、それぞれ1、0に初期化される。

[0143] このように、W＝1のウェイが存在しない場合、リプレース対象は、使用フラグUが0のキャッシュエントリーの中から1つ選択される。

[0144] また、W＝1のウェイが存在する場合、リプレース対象は、使用フラグUが0であると1であるとを問わず、W＝1のウェイのキャッシュエントリーから1つ選択される。これにより図14(a)(b)に示したように、アクセス頻度の低いデータがキャッシュメモリに残ることによるキャッシュミスの誘発を低減することができる。

[0145] 以上説明してきたように、本実施の形態におけるキャッシュメモリによれば、プロセッサ1によるストア命令の進行状態から、ストアが完了してラインを推定し、推定したラインに対してCフラグ又はWフラグを設定するので、ストアが完了した直前のラインを効率よくリプレース対象を指定することができる。その際、プロセッサ1によってキャッシュメモリのライン境界やラインサイズを管理する必要がなく、キャッシュ管理のための負荷を小さくすることができる。

[0146] また、ウィークフラグW＝1でかつダーティフラグ＝1のラインを、プロセッサからこれ以上書き込みがなされないラインとして、クリーニングすることにより、キャッシュミス時のライトバックペナルティを低減することができる。

[0147] また、これ以上使用されないキャッシュエントリーにW＝1が設定され、W＝1のキャッシュエントリーが真っ先にリプレース対象として選択されるので、アクセス頻度の低いデータがキャッシュメモリに残ることによるキャッシュミスの誘発を低減することができる。

[0148] また、従来のLRU方式におけるアクセス順序を示すデータの代わりに1ビットの使用フラグを用いる擬似LRU方式を採用することにより、アクセス順序データとして1ビットのフラグでよいので、アクセス順序データのデータ量が少ないこと及び更新が簡単であることからハードウェア規模を小さくすることができる。

[0149] ＜変形例＞

(1)図4(a)(b)(c)に示した各命令は、コンパイラによりプログラム中に挿入してもよい。その際、コンパイラは、例えば配列データの書き込みや、圧縮動画データをデコードする際のブロックデータの書き込み等、これ以上書き込みをしないプログラム位置に、上記各命令を挿入するようにすればよい。

(2)上記クリーニング処理において無効化処理を行う構成としてもよい。すなわち、図17に示したフローチャートにおいて、S907にてダーティでないと判定された場合、さらに、当該キャッシュエントリーを無効化(Vフラグをリセット)するステップを追加してもよい。さらに、クリーニング処理におけるライトバックの後に無効化するステップを追加してもよい。

産業上の利用可能性

[0150] 本発明は、メモリアクセスを高速化するためのキャッシュメモリに適しており、例えば、オンチップキャッシュメモリ、オフチップキャッシュメモリ、データキャッシュメモリ、命令キャッシュメモリ等に適している。

請求の範囲

[1] プロセッサの状態に関する条件を生成する条件生成手段と、

現在のプロセッサの状態が前記条件を満たすかどうかを判定する判定手段と、

操作対象となるアドレスを生成するアドレス生成手段と、

前記判定手段が条件を満たすと判定したときに前記アドレス生成手段によって生成されたアドレスを用いてキャッシュを操作する操作手段と

を備えることを特徴とするキャッシュメモリシステム。

[2] 前記条件生成手段は、前記判定手段が条件を満たすと判定した場合に新たな条件を生成する

ことを特徴とする請求項1記載のキャッシュメモリシステム。

[3] 前記条件生成手段は、プロセッサ内の特定レジスタの値に関する条件を生成する

ことを特徴とする請求項2記載のキャッシュメモリシステム。

[4] 前記特定レジスタはプログラムカウンタである

ことを特徴とする請求項3記載のキャッシュメモリシステム。

[5] 前記条件生成手段は、特定のアドレス範囲内へのメモリアクセスおよび特定のアドレス範囲外へのメモリアクセスの何れかを前記条件として生成する

ことを特徴とする請求項2記載のキャッシュメモリシステム。

[6] 前記条件生成手段は、プロセッサが特定命令を実行することを前記条件として生成する

ことを特徴とする請求項1記載のキャッシュメモリシステム。

[7] 前記条件生成手段は、現在の条件に特定の演算を施すことによって前記新たな条件を生成する

ことを特徴とする請求項2記載のキャッシュメモリシステム。

[8] 前記条件生成手段はメモリアクセスアドレスを条件として生成し、

前記判定手段が条件を満たすと判定した場合に現在の条件に定数を加算することによって前記新たな条件を生成する

ことを特徴とする請求項7記載のキャッシュメモリシステム。

[9] 前記定数は、プロセッサにより実行されるポストインクリメント付きロード／ストア命令

におけるインクリメント値またはデクリメント値、およびプロセッサにより実行される2回のロード／ストア命令におけるアドレスの差分値の何れかである

ことを特徴とする請求項8記載のキャッシュメモリシステム。

[10] 前記条件生成手段は複数の条件を生成し、

前記判定手段は、複数の条件のすべてを満たすかどうかを判定する

ことを特徴とする請求項1記載のキャッシュメモリシステム。

[11] 前記条件生成手段は複数の条件を生成し、

前記判定手段は、複数の条件の何れかを満たすかどうかを判定する

ことを特徴とする請求項1記載のキャッシュメモリシステム。

[12] 前記操作手段は、

前記判定手段が条件を満たすと判定したときに、前記アドレス生成手段により生成されたアドレスに対応するデータがキャッシュに格納されているかどうかを判定するデータ判定手段と、

格納されていないと判定された場合に、キャッシュメモリ中のラインを選択する選択手段と、

前記選択されたラインが有効でダーディならライトバックを行うライトバック手段と、

前記アドレスに対応するデータをメモリからライトバック後の選択されたラインへ転送する転送手段と、

前記アドレスをタグとして前記選択されたラインへ登録する登録手段と

を備えることを特徴とする請求項1から3の何れかに記載のキャッシュメモリシステム。

[13] 前記操作手段は、

前記判定手段が条件を満たすと判定したときに、前記アドレス生成手段により生成されたアドレスに対応するデータがキャッシュに格納されているかどうかを判定するデータ判定手段と、

格納されていないと判定された場合に、キャッシュメモリ中のラインを選択する選択手段と、

選択されたラインが有効でダーティであれば、ライトバックを行うライトバック手段と、

メモリから選択されたラインへデータを転送することなく、前記生成したアドレスをタグとして選択されたラインへ登録する登録手段と

を備えることを特徴とする請求項1から3の何れかに記載のキャッシュメモリシステム。

[14] 前記操作手段は、

前記判定手段が条件を満たすと判定したときに、前記アドレス生成手段により生成されたアドレスに対応するデータがキャッシュに格納されているかどうかを判定するデータ判定手段と、

格納されていると判定された場合に、キャッシュメモリ中の格納先のラインを選択する選択手段と、

選択されたラインが有効でかつダーディであればライトバックを行うライトバック手段と、

を備えることを特徴とする請求項1から3の何れかに記載のキャッシュメモリシステム。

[15] 前記操作手段は、

前記判定手段が条件を満たすと判定したときに、前記アドレス生成手段により生成されたアドレスに対応するデータがキャッシュに格納されているかどうかを判定するデータ判定手段と、

格納されていると判定された場合に、キャッシュメモリ中の格納先のラインを選択する選択手段と、

選択されたラインを無効化する無効化手段と

を備えることを特徴とする請求項1から3の何れかに記載のキャッシュメモリシステム。

[16] 前記操作手段は、

前記判定手段が条件を満たすと判定したときに、前記アドレス生成手段により生成されたアドレスに対応するデータがキャッシュに格納されているかどうかを判定するデータ判定手段と、

格納されていると判定された場合に、キャッシュメモリ中の格納先のラインを選択す

る選択手段と、

ラインのアクセス順序を示す順序情報に対して、選択されたラインのアクセス順序を変更する変更手段と、

を備えることを特徴とする請求項1から3の何れかに記載のキャッシュメモリシステム。

[17] 前記条件生成手段により前記条件としてメモリアドレスを生成し、

前記操作手段は、さらに、

前記条件生成手段により生成されたメモリアドレスがラインの途中を指す場合に、当該ラインの先頭、次のラインの先頭および前のラインの先頭の何れかを指すように調整することによりアドレスを生成する調整手段を備える

ことを特徴とする請求項12から16の何れかに記載ののキャッシュシステム。

[18] キャッシュメモリの制御方法であって、

プロセッサの状態に関する条件を生成する条件生成ステップと、

現在のプロセッサの状態が前記条件を満たすかどうかを判定する判定ステップと、

操作対象となるアドレスを生成するアドレス生成ステップと、

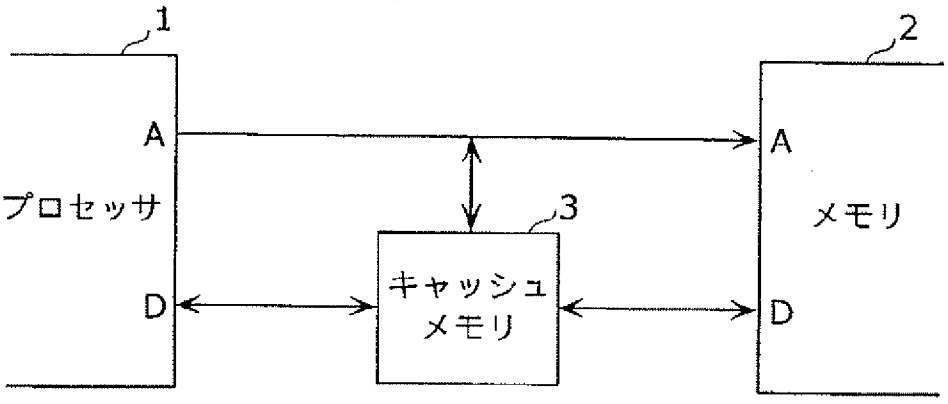
前記判定ステップにおいて条件を満たすと判定したときに前記アドレス生成ステップにおいて生成されたアドレスを用いてキャッシュを操作する操作ステップと

を有することを特徴とする制御方法。

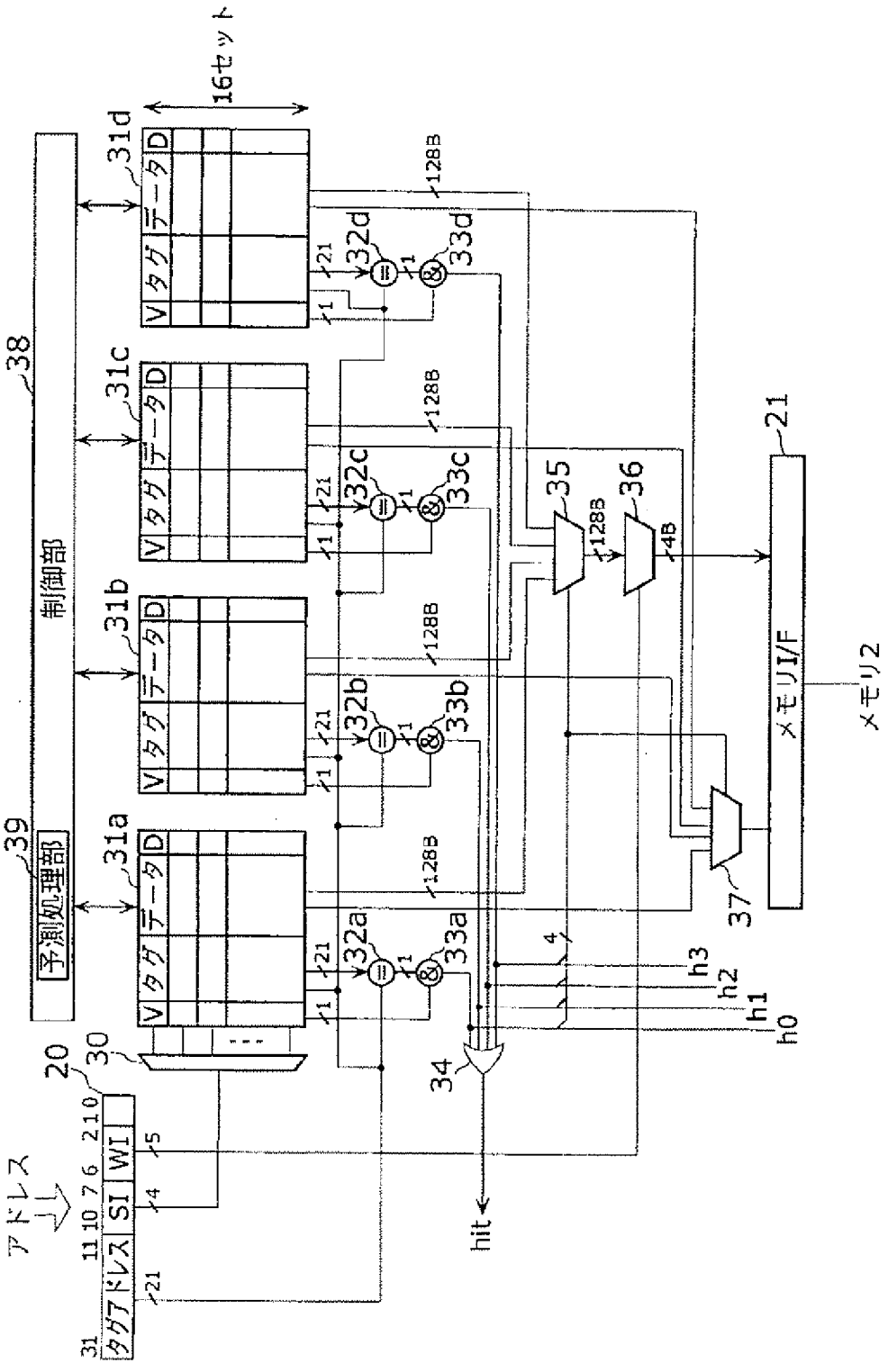
要 約 書

本発明のキャッシュメモリは、プロセッサから出力されるメモリアクセスの進行状況に基づいて次にプリフェッチすべきラインアドレスを予測する予測処理部39を有し、予測処理部39は、メモリからキャッシュメモリに予測されたラインアドレスのデータをプリフェッチするプリフェッチ部414と、メモリからキャッシュメモリにデータをロードすることなく、予測されたラインアドレスをタグとしてキャッシュエントリーに設定し、バリッドフラグを有効にするタッチ部415とを備える。

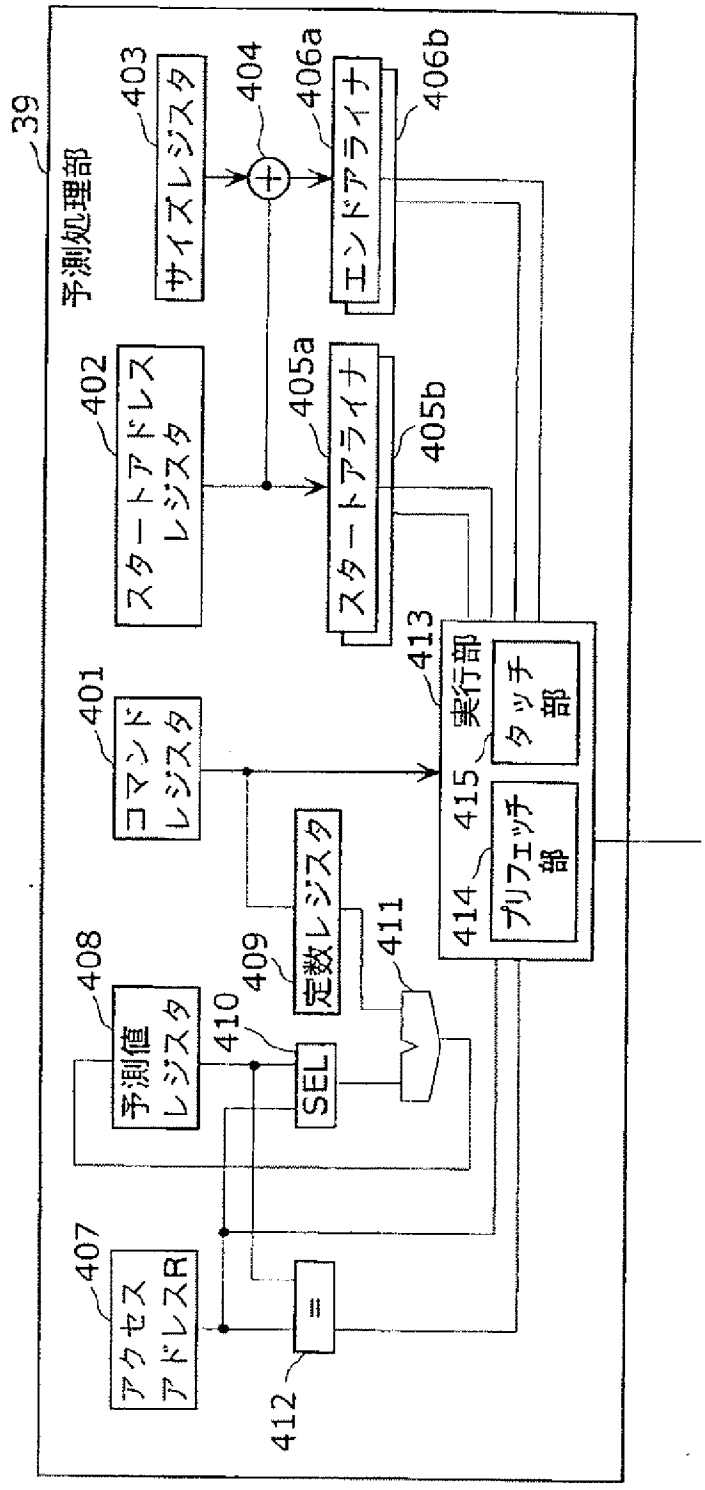
[図1]



[図2]

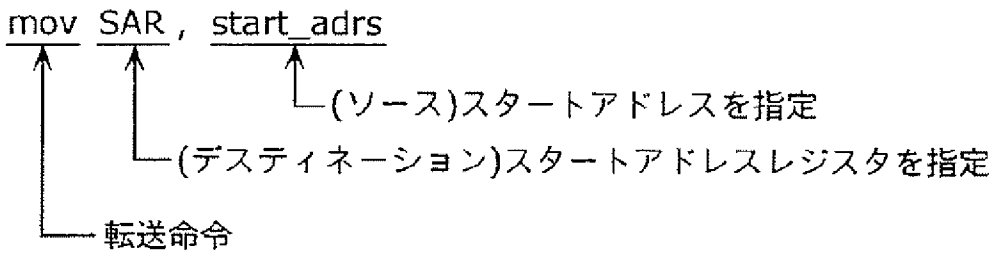

16セット
31d
31c
31b
31a
V タグ データ D
128B
32d
32c
32b
32a
33d
33c
33b
33a
21
1
38
制御部
39
予測処理部
20
30
アドレス
11 10 7 6 2 1 0
SI WI
タグアドレス
31
4
5
34
35
36
37
4B
h3
h2
h1
h0
hit
メモリI/F
メモリ2

[図3]

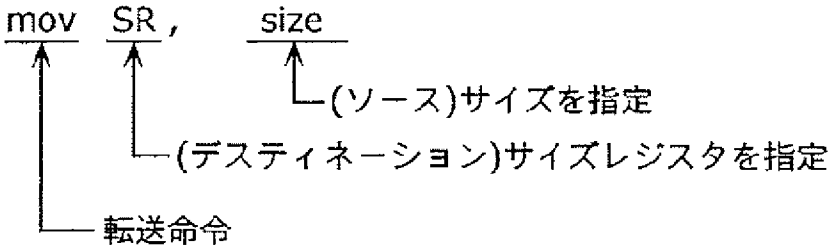

39
予測処理部
403
サイズレジスタ
404
406a
エンドアライナ
406b
402
スタートアドレス
レジスタ
405a
スタートアライナ
405b
413
実行部
タッチ
部
401
コマンド
レジスタ
415
プリフェッチ
部
414
408
予測値
レジスタ
409
定数レジスタ
411
410
SEL
407
アクセス
アドレスR
=
412

[図4]

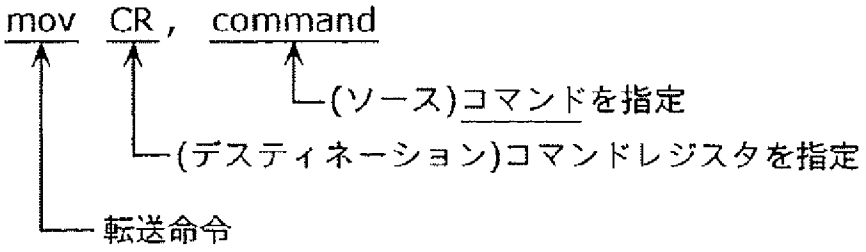
(a)


mov SAR , start_adrs
(ソース)スタートアドレスを指定
(デスティネーション)スタートアドレスレジスタを指定
転送命令

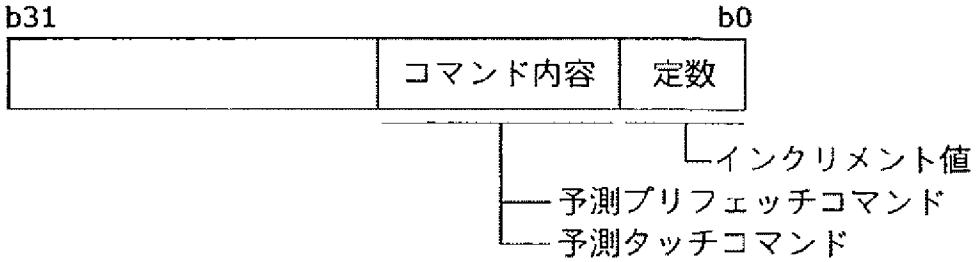
(b)


mov SR , size
(ソース)サイズを指定
(デスティネーション)サイズレジスタを指定
転送命令

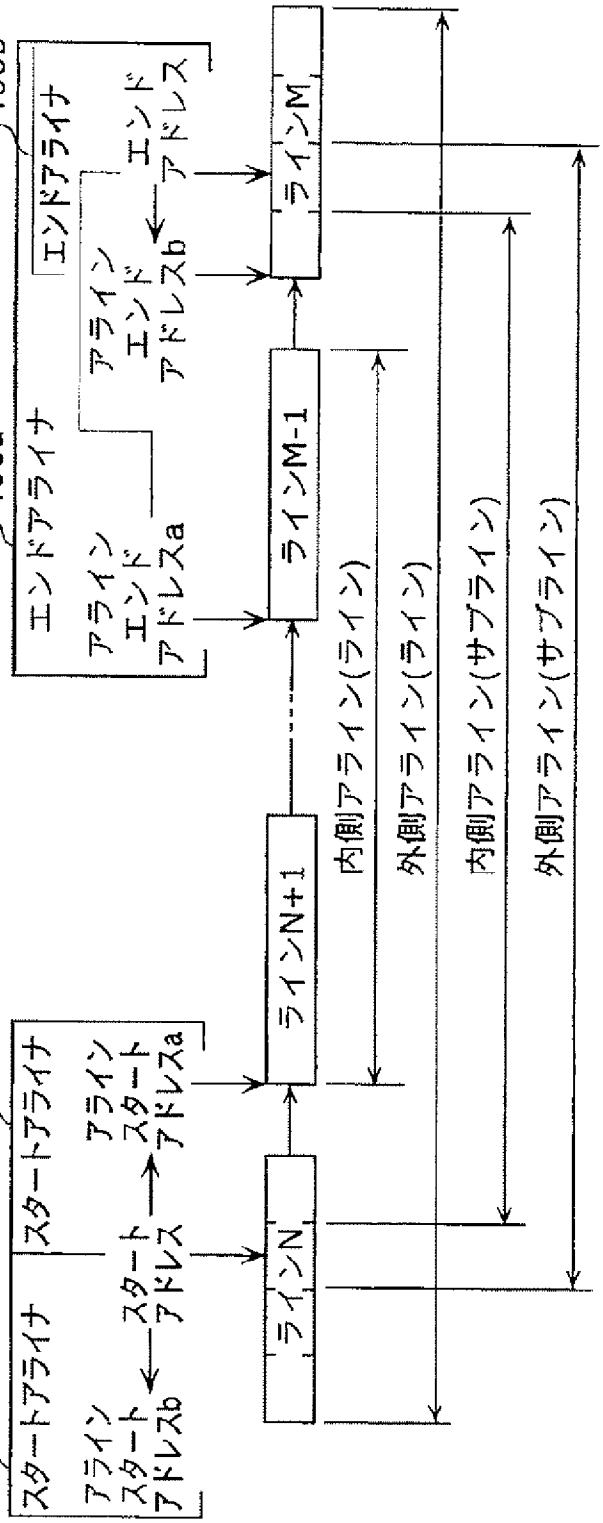
(c)


mov CR , command
(ソース)コマンドを指定
(デスティネーション)コマンドレジスタを指定
転送命令

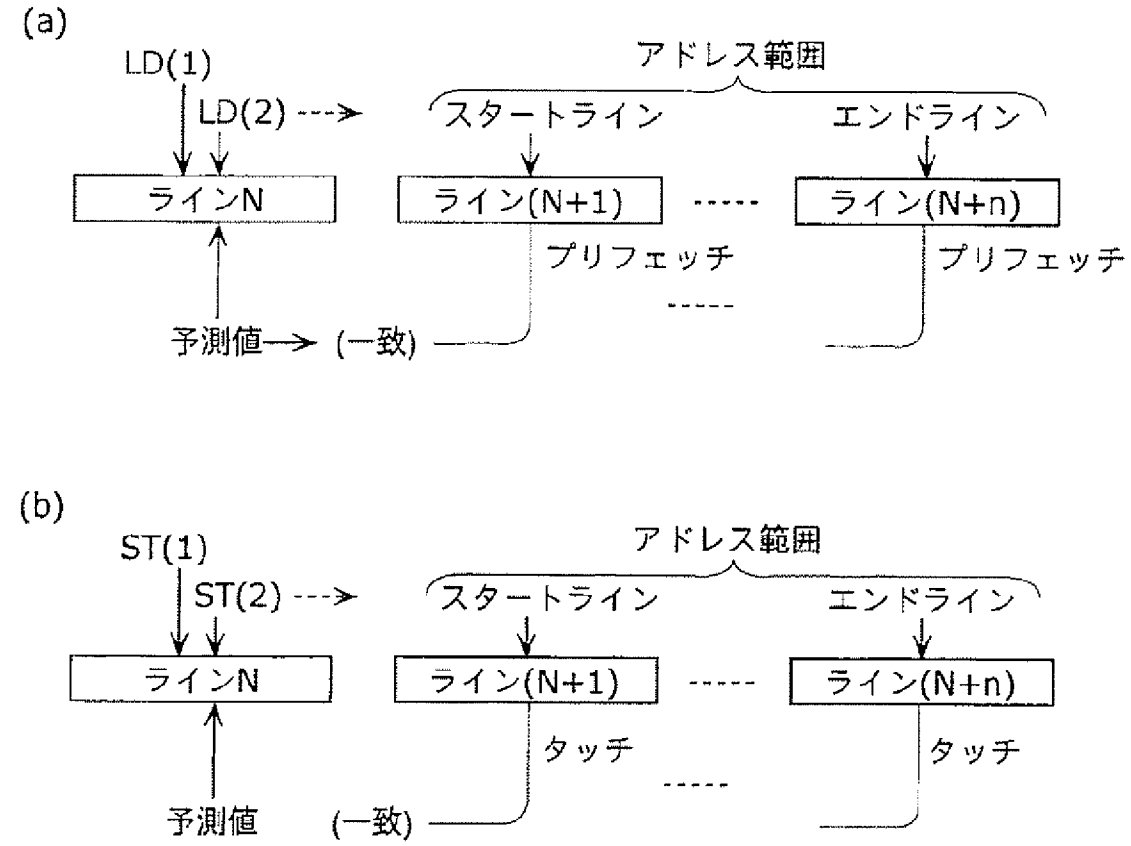
(d)


b31
b0
コマンド内容
定数
インクリメント値
予測プリフェッチコマンド
予測タッチコマンド

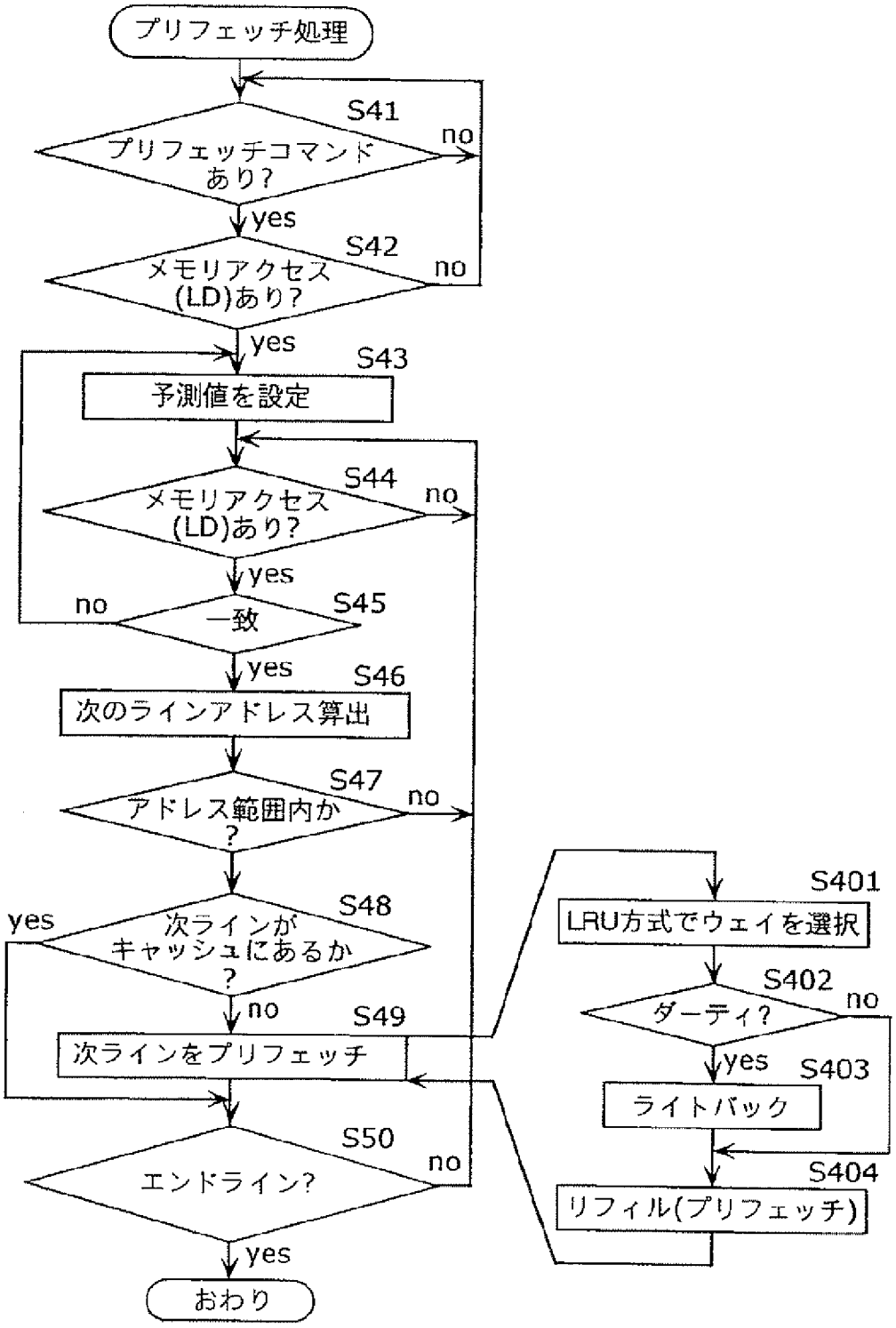
[図5]


405b
スタートアライナ
アライン
スタート
アドレスb
405a
スタートアライナ
アライン
スタート
アドレスa
スタート
アドレス
ラインN
ラインN+1
406a
エンドアライナ
アライン
エンド
アドレスa
406b
エンドアライナ
アライン
エンド
アドレスb
エンド
アドレス
ラインM-1
ラインM
内側アライン(ライン)
外側アライン(ライン)
内側アライン(サブライン)
外側アライン(サブライン)

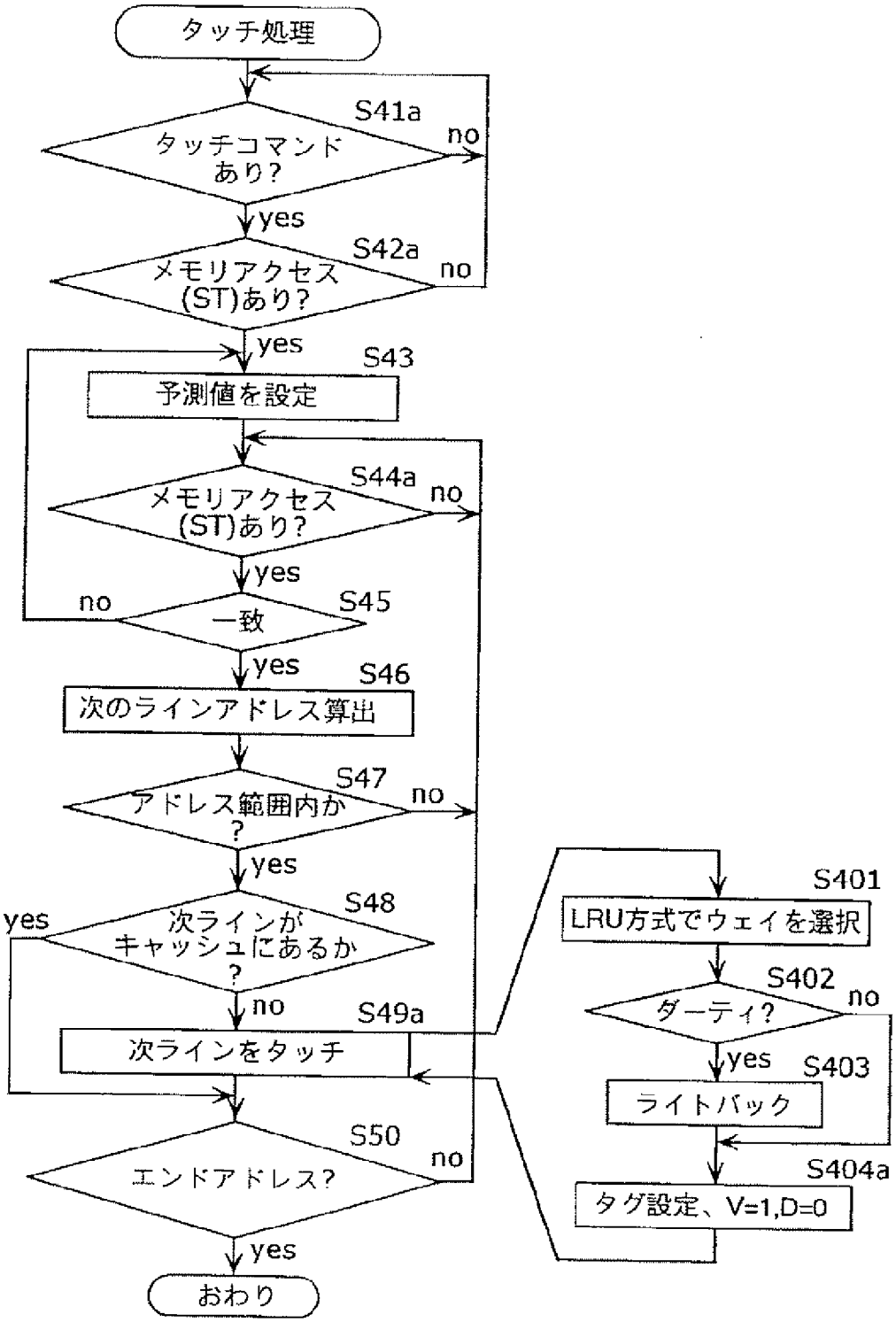
[図6]


(a)
LD(1)
LD(2)
アドレス範囲
スタートライン
エンドライン
ラインN
ライン(N+1)
ライン(N+n)
プリフェッチ
プリフェッチ
予測値
(一致)
(b)
ST(1)
ST(2)
アドレス範囲
スタートライン
エンドライン
ラインN
ライン(N+1)
ライン(N+n)
タッチ
タッチ
予測値
(一致)

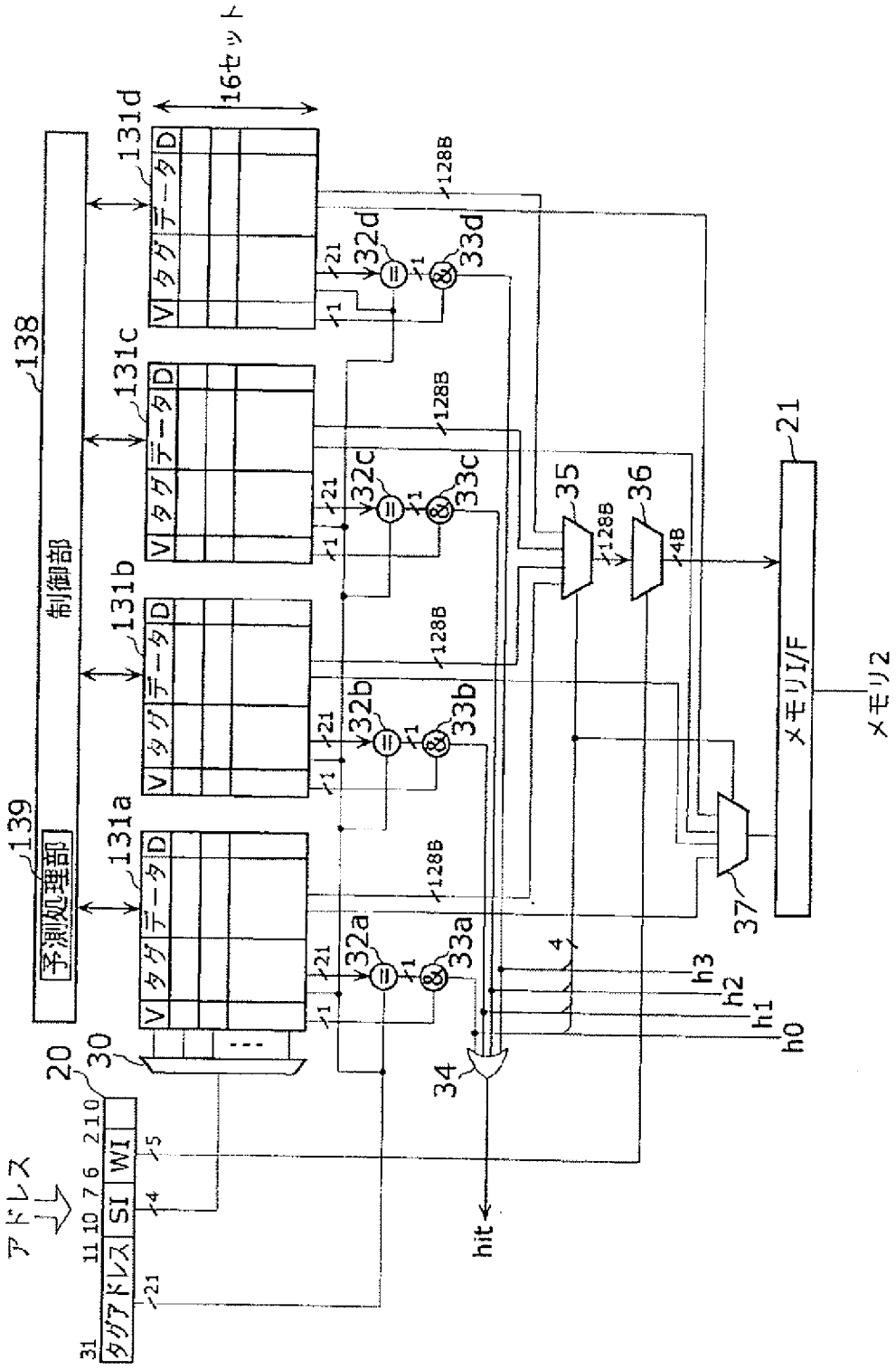
[図7]


プリフェッチ処理
S41
プリフェッチコマンドあり?
no
yes
S42
メモリアクセス(LD)あり?
no
yes
S43
予測値を設定
S44
メモリアクセス(LD)あり?
no
yes
no
S45
一致
yes
S46
次のラインアドレス算出
S47
アドレス範囲内か?
no
S48
yes
次ラインがキャッシュにあるか?
no
S49
次ラインをプリフェッチ
S50
エンドライン?
no
yes
おわり
S401
LRU方式でウェイを選択
S402
ダーティ?
no
yes
S403
ライトバック
S404
リフィル(プリフェッチ)

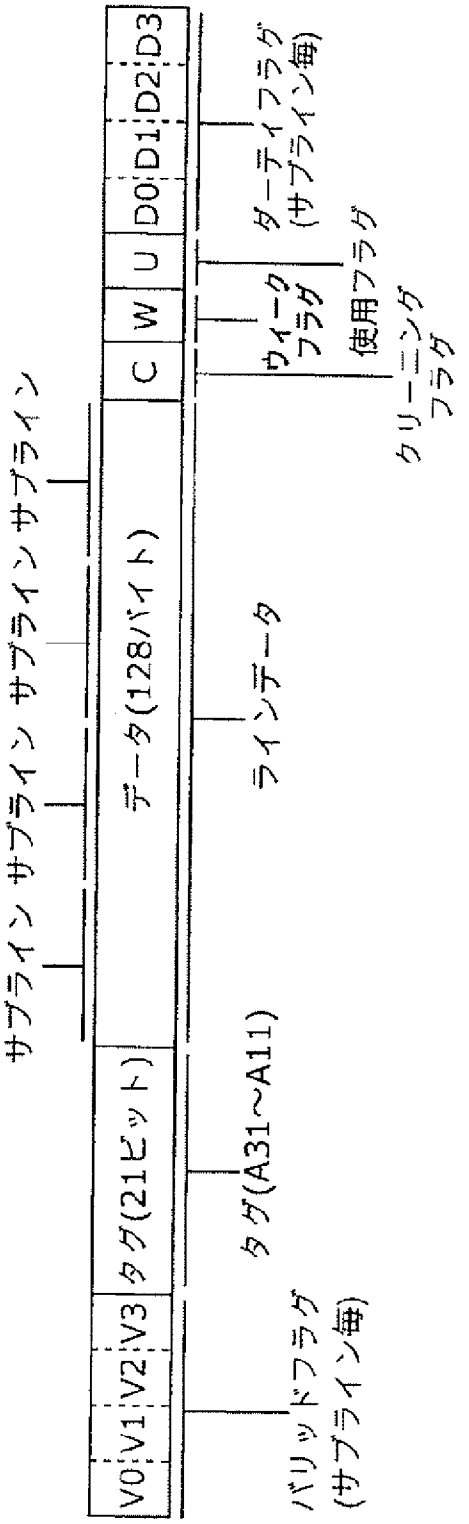
[図8]


タッチ処理
S41a
タッチコマンド
あり?
no
yes
S42a
メモリアクセス
(ST)あり?
no
yes
S43
予測値を設定
S44a
メモリアクセス
(ST)あり?
no
yes
no
S45
一致
yes
S46
次のラインアドレス算出
S47
アドレス範囲内か
?
no
yes
yes
S48
次ラインが
キャッシュにあるか
?
no
S49a
次ラインをタッチ
S50
エンドアドレス?
no
yes
おわり
S401
LRU方式でウェイを選択
S402
ダーティ?
no
yes
S403
ライトバック
S404a
タグ設定、V=1,D=0

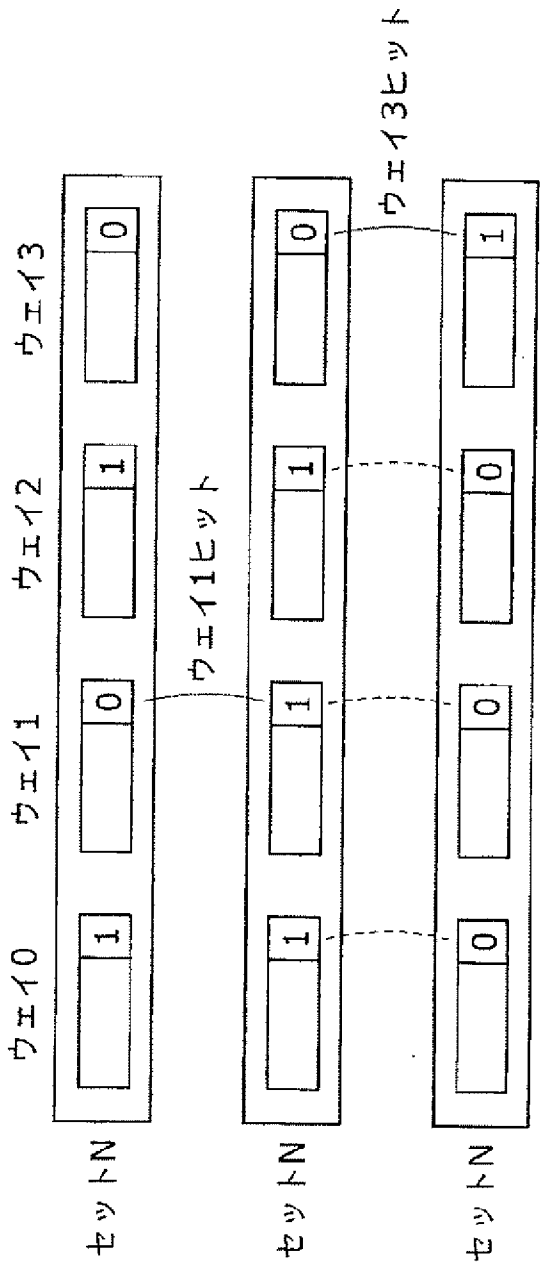
[図9]


16セット
131d
131c
131b
131a
138
制御部
139
予測処理部
20
30
アドレス
タグアドレス
SI
WI
V
タグ
データ
D
128B
4B
32a
32b
32c
32d
33a
33b
33c
33d
34
35
36
37
21
メモリI/F
メモリ2
hit
h0
h1
h2
h3

[図10]

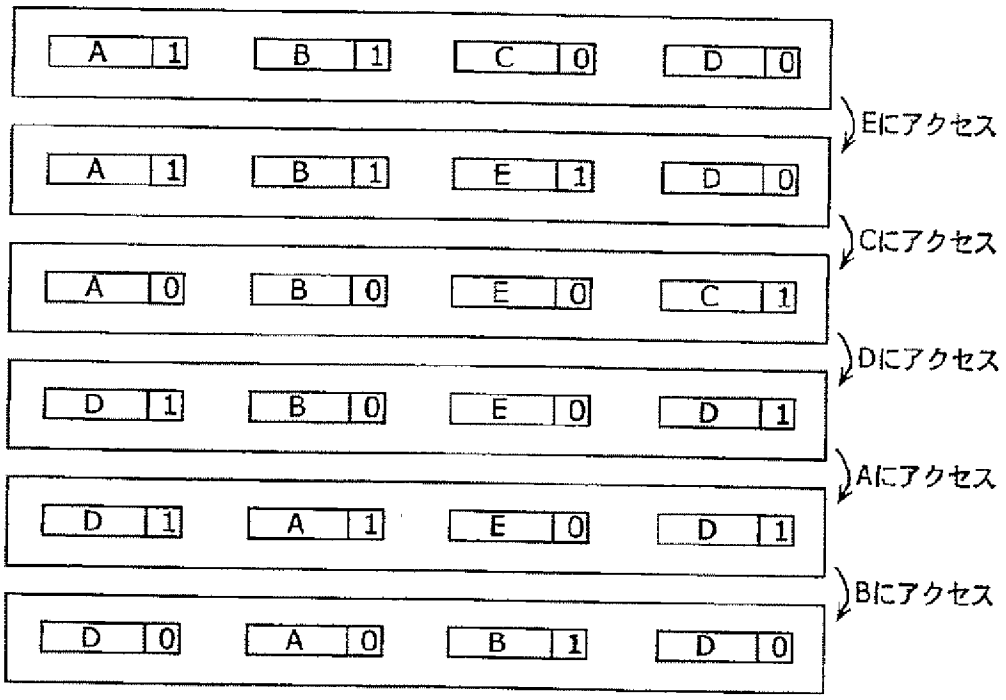

サブライン
サブライン
サブライン
サブライン
サブライン
V0
V1
V2
V3
タグ(21ビット)
データ(128バイト)
C
W
U
D0
D1
D2
D3
パリッドフラグ
(サブライン毎)
タグ(A31~A11)
ラインデータ
クリーニング
フラグ
ウィーク
フラグ
使用フラグ
ダーティフラグ
(サブライン毎)

[図11]

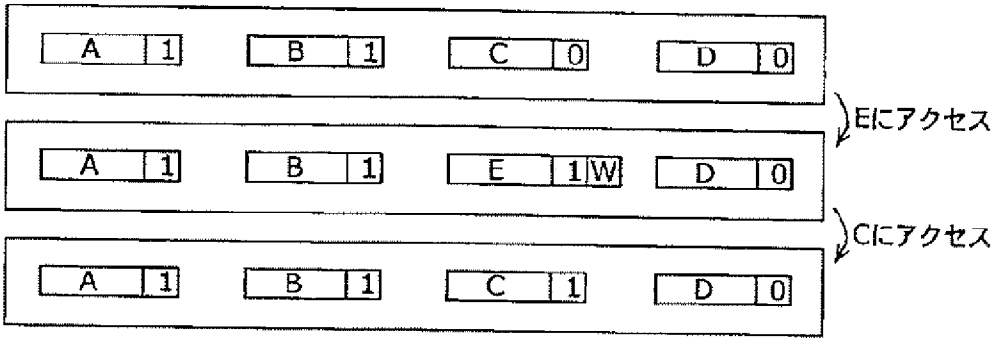


[図12]

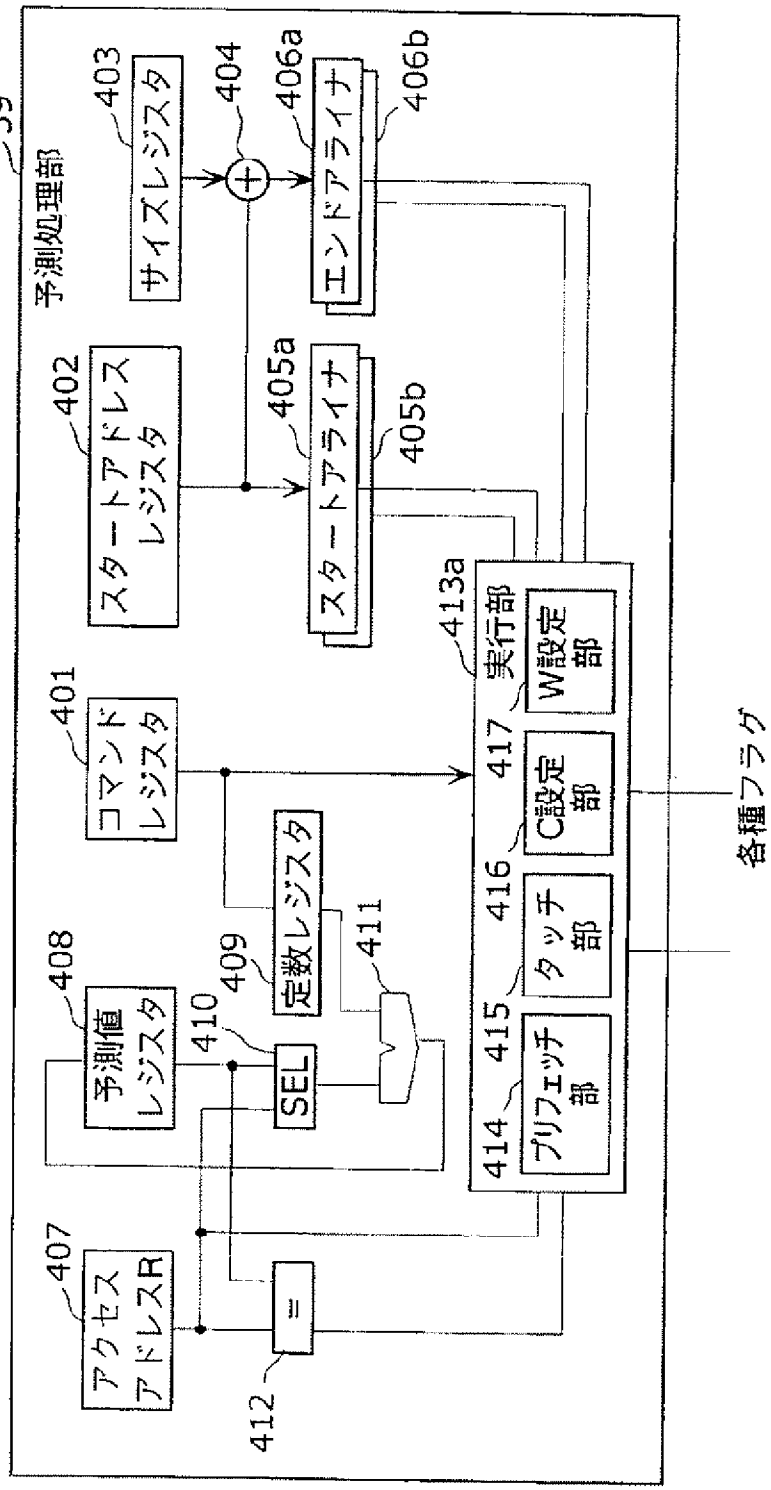
(a)


A 1
B 1
C 0
D 0
Eにアクセス
A 1
B 1
E 1
D 0
Cにアクセス
A 0
B 0
E 0
C 1
Dにアクセス
D 1
B 0
E 0
D 1
Aにアクセス
D 1
A 1
E 0
D 1
Bにアクセス
D 0
A 0
B 1
D 0

(b)

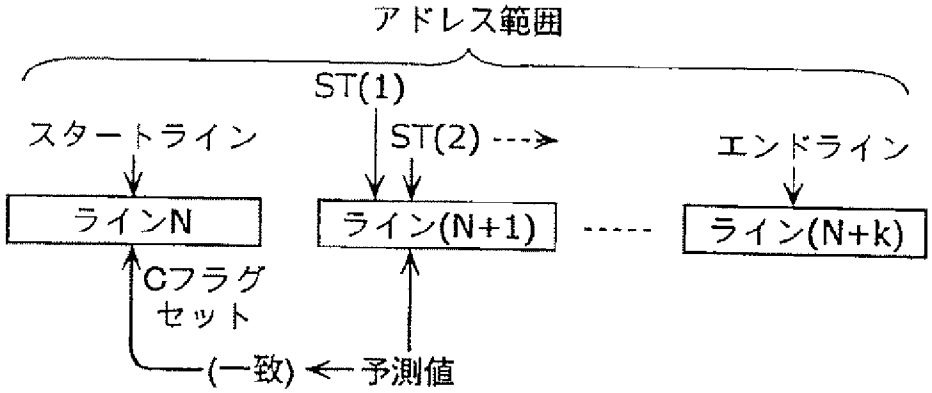

A 1
B 1
C 0
D 0
Eにアクセス
A 1
B 1
E 1 W
D 0
Cにアクセス
A 1
B 1
C 1
D 0

[図13]

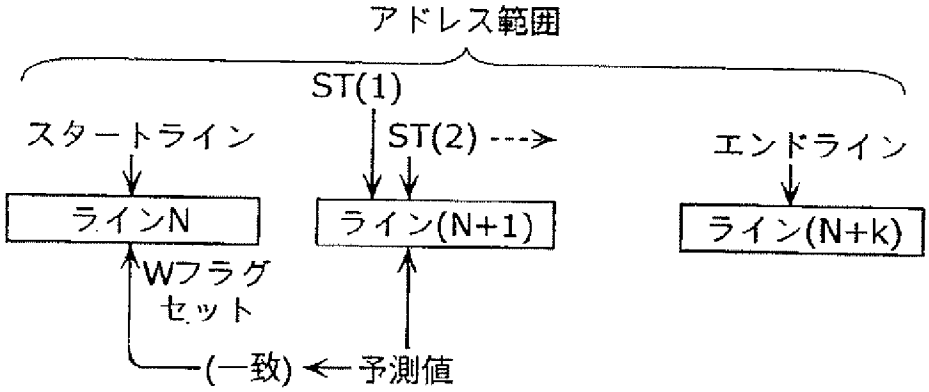

39
予測処理部
403
サイズレジスタ
404
406a
エンドアライナ
406b
402
スタートアドレス
レジスタ
405a
スタートアライナ
405b
413a
実行部
W設定
部
401
コマンド
レジスタ
417
C設定
部
各種フラグ
416
タッチ
部
408
予測値
レジスタ
409
定数レジスタ
410
411
SEL
415
プリフェッチ
部
414
407
アクセス
アドレスR
412
=

[図14]

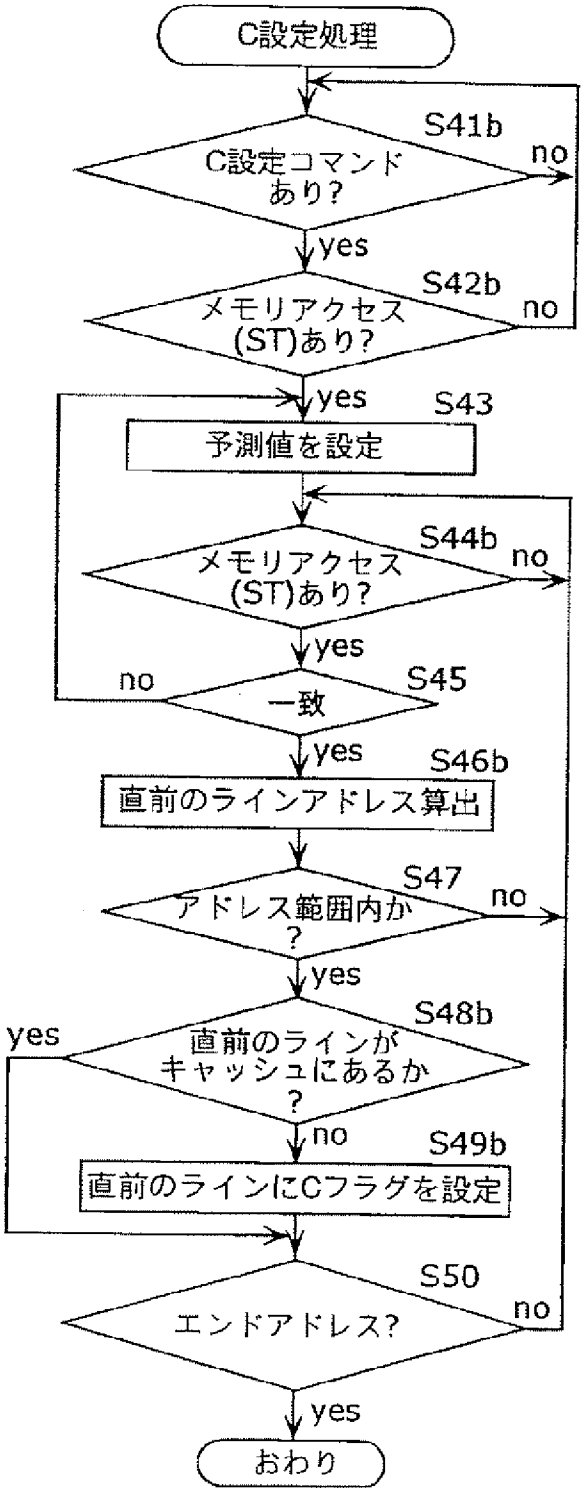
(a)


アドレス範囲
ST(1)
スタートライン
ST(2)
エンドライン
ラインN
ライン(N+1)
ライン(N+k)
Cフラグ
セット
(一致)
予測値

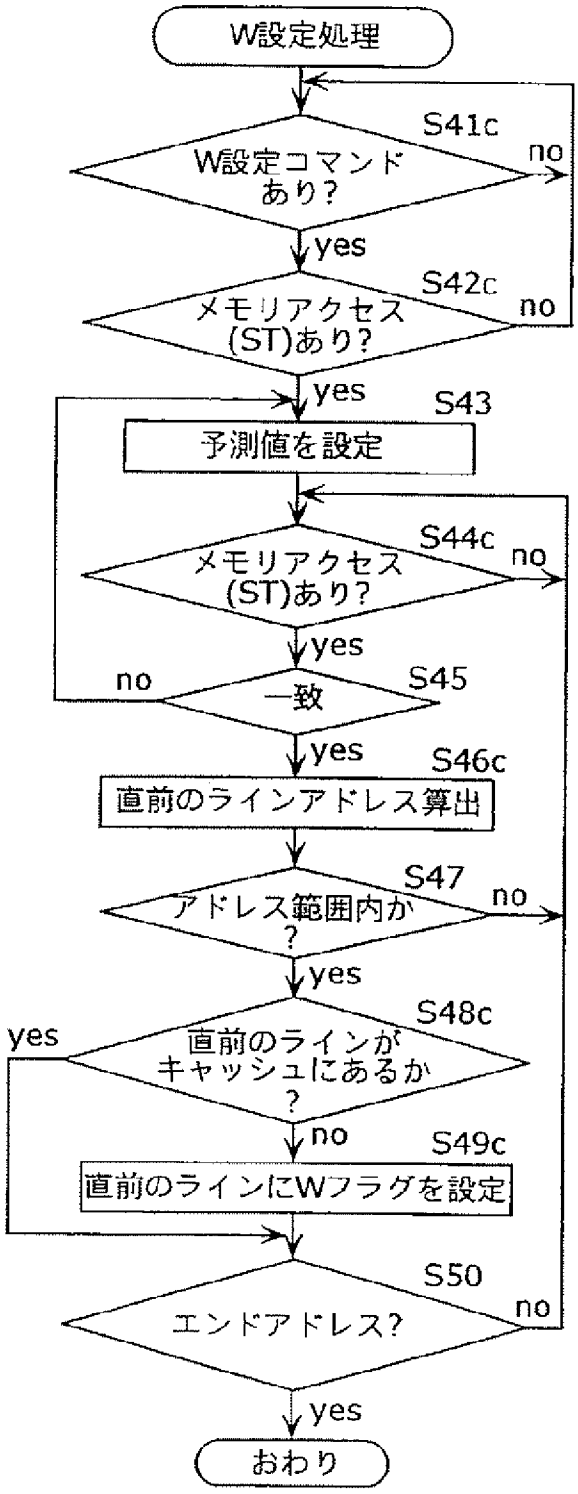
(b)


アドレス範囲
ST(1)
スタートライン
ST(2)
エンドライン
ラインN
ライン(N+1)
ライン(N+k)
Wフラグ
セット
(一致)
予測値

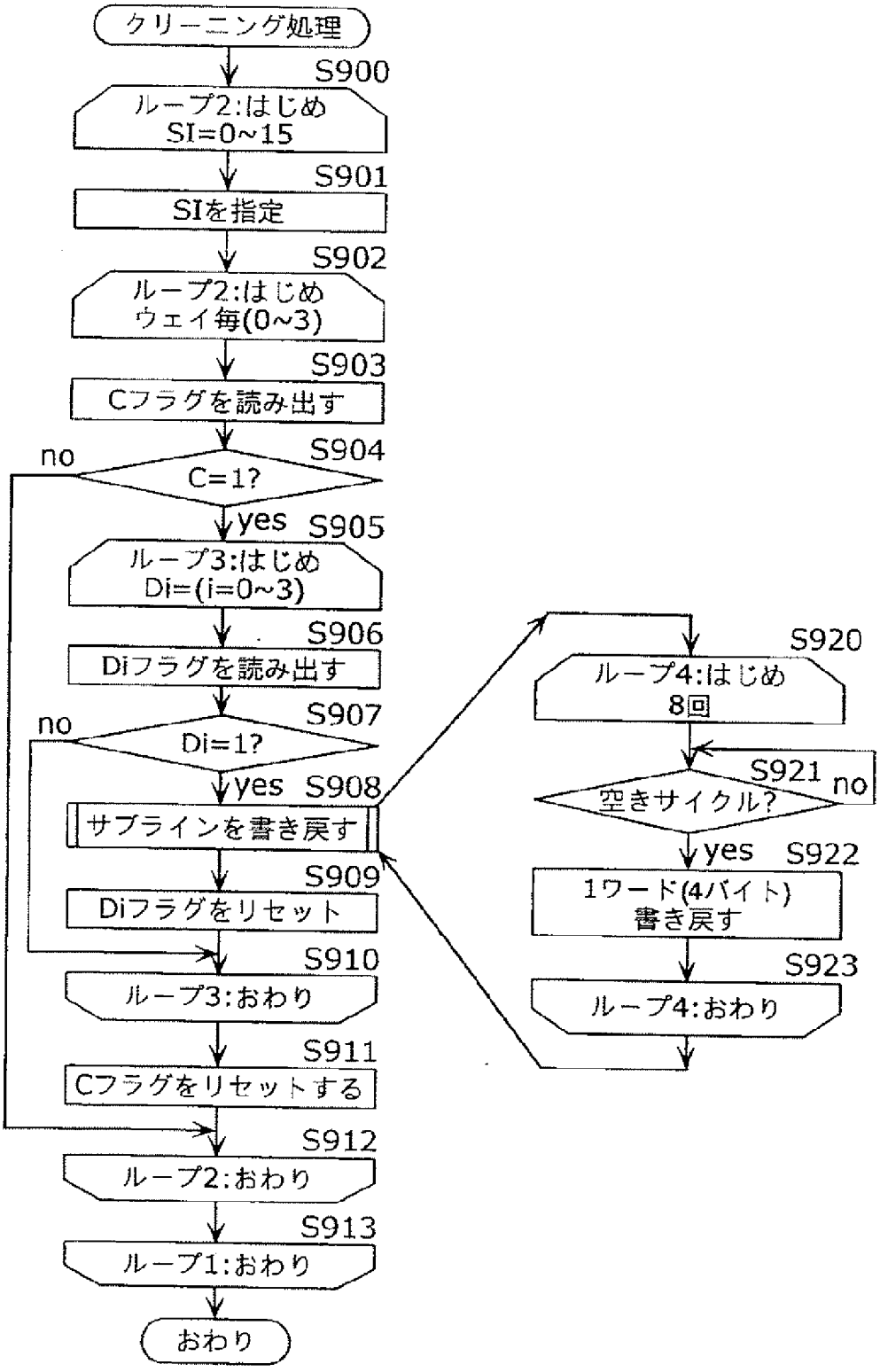
[図15]


C設定処理
S41b
C設定コマンドあり?
no
yes
S42b
メモリアクセス(ST)あり?
no
yes
S43
予測値を設定
S44b
メモリアクセス(ST)あり?
no
yes
no
S45
一致
yes
S46b
直前のラインアドレス算出
S47
アドレス範囲内か?
no
yes
S48b
直前のラインがキャッシュにあるか?
yes
no
S49b
直前のラインにCフラグを設定
S50
エンドアドレス?
no
yes
おわり

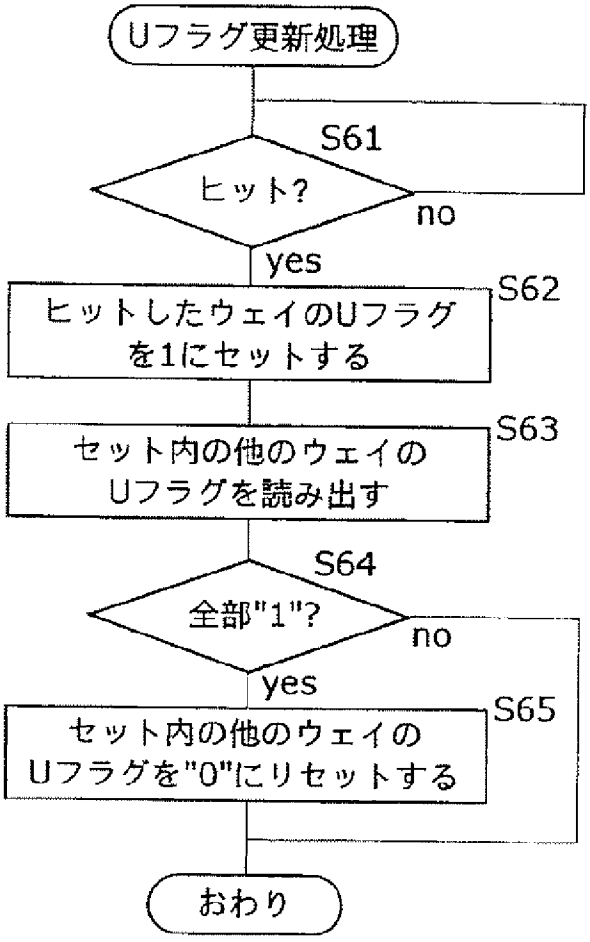
[図16]


W設定処理
S41c
W設定コマンドあり?
no
yes
S42c
メモリアクセス(ST)あり?
no
yes
S43
予測値を設定
S44c
メモリアクセス(ST)あり?
no
yes
no
S45
一致
yes
S46c
直前のラインアドレス算出
S47
アドレス範囲内か?
no
yes
S48c
直前のラインがキャッシュにあるか?
yes
no
S49c
直前のラインにWフラグを設定
S50
エンドアドレス?
no
yes
おわり

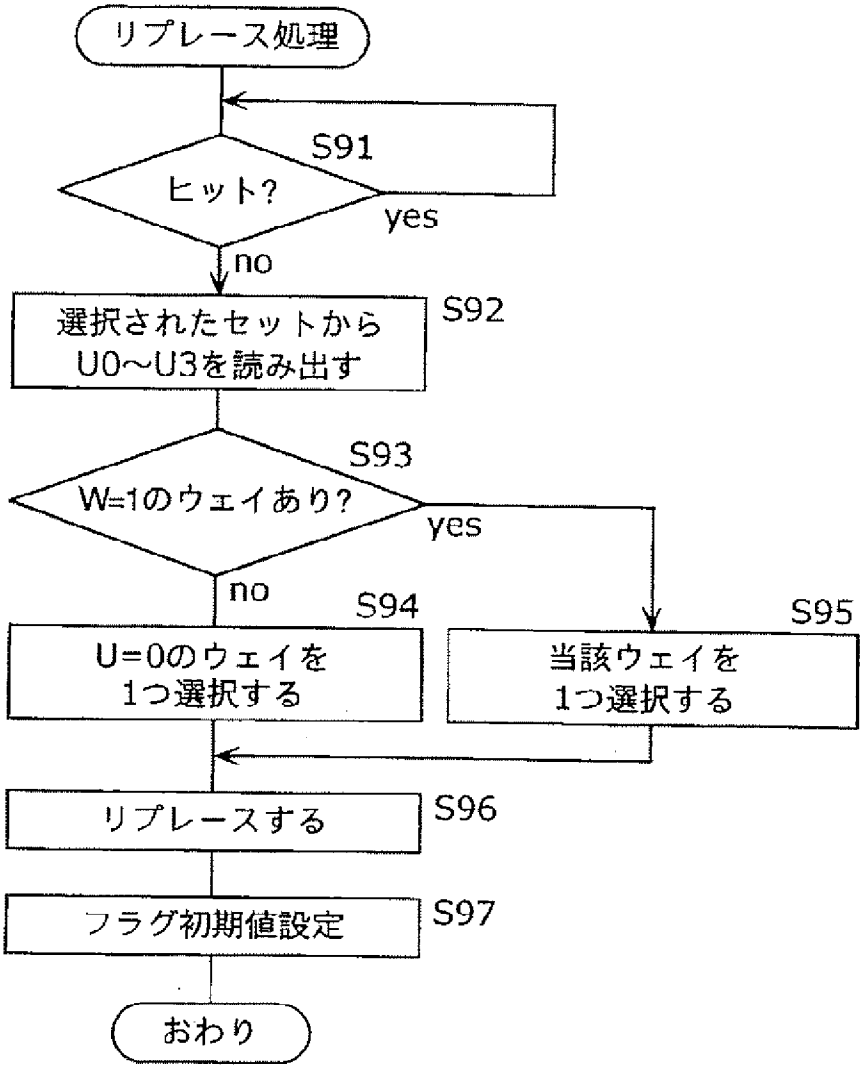
[図17]


クリーニング処理
S900
ループ2:はじめ
SI=0~15
S901
SIを指定
S902
ループ2:はじめ
ウェイ毎(0~3)
S903
Cフラグを読み出す
S904
C=1?
no
yes
S905
ループ3:はじめ
Di=(i=0~3)
S906
Diフラグを読み出す
S907
Di=1?
no
yes
S908
サブラインを書き戻す
S909
Diフラグをリセット
S910
ループ3:おわり
S911
Cフラグをリセットする
S912
ループ2:おわり
S913
ループ1:おわり
おわり
S920
ループ4:はじめ
8回
S921
空きサイクル?
no
yes
S922
1ワード(4バイト)
書き戻す
S923
ループ4:おわり

[図18]


Uフラグ更新処理
S61
ヒット?
no
yes
S62
ヒットしたウェイのUフラグ
を1にセットする
S63
セット内の他のウェイの
Uフラグを読み出す
S64
全部"1"?
no
yes
S65
セット内の他のウェイの
Uフラグを"0"にリセットする
おわり

[図19]


リプレース処理
S91
ヒット?
yes
no
選択されたセットから
U0~U3を読み出す
S92
S93
W=1のウェイあり?
yes
no
S94
U=0のウェイを
1つ選択する
S95
当該ウェイを
1つ選択する
リプレースする
S96
フラグ初期値設定
S97
おわり